Programming Environment for Management of Parallel Jobs with Concurrent Access to a Common Data Source

Ivaylo Penev[1]

Abstract – **The paper presents a strategy and a programming environment for effective realization of parallel jobs with concurrent access to a common data source. Each job is started on a separate workstation in a local computer network. In order to avoid the conflicts, caused by the simultaneous access to the data source and to achieve better performance time, an approach with delay start of the jobs is proposed.**

Keywords – **Parallel jobs, Concurrent jobs, Simulation problems.**

I. INTRODUCTION

In the area of banking information systems a class of problems, associated with simulation of many financial portfolios exists. In general each problem passes through reading data about portfolios from a data source (for example database), executing simulation calculations according to the performed analyses, and storing results in the same data source. The calculating procedures about each portfolio are completely independent each other. The whole simulation process is divided to jobs, each one simulating a separate portfolio. Prerequisites about realizations of such a class of problems in parallel computing environments exist. The efficiency of the realizations is significantly decreased due to the concurrent access of many parallel jobs to the common resource. The current work presents a strategy for preventing the concurrent access by proper shifting of jobs along the time axis of execution. The strategy is practically realized within a programming environment for management of the parallel jobs.

II. REALIZATION OF A PORTFOLIO MANAGEMENT SYSTEM IN A DISTRIBUTED COMPUTING ENVIRONMENT

The researches, presented in the current work, are based on the realization of a Portfolio management system (PMS) in a distributed computing environment [2]. The system simulates financial portfolios, calculating their positions. Information about each portfolio is stored in a database. When simulation about a specific portfolio is started, relevant data from the database are retrieved and suitable calculations are performed. The simulation finishes with storing report data in the database. Because of the great number of portfolios, positions and the various analyses performed, the simulation is heavy calculating process, taking long time to be finished on a single computer. Therefore the PMS executable is started on multiple connected computers, each of them performing simulation of a separate portfolio and saving the results in the database.

For managing the environment of computers (calculating resources, nodes) the system Condor is used, developed and spread for free by the University of Wisconsin-Madison, USA [3]. The distributed environment consists of a pool of computers and a server for database management (Oracle server), storing data about financial portfolios.

Each job has a description (in a submit description file) matching a portfolio simulation with a computer from the pool.

An execution of four jobs in series by a single computer and in parallel by the presented environment is tested. The time results differ from the expected. The increase of the count of the computers, participating in the simulation, decreases the parallel execution efficiency. The database management server is unable to process lots of simultaneous queries. Therefore, the concurrent access of the jobs to the common resource is a strong limiting condition over the execution time of the whole process. Applying a strategy for avoiding this limitation is necessary.

III. STRATEGY FOR MINIMIZING THE CONCURRENT ACCESS OF PARALLEL JOBS TO A COMMON DATA SOURCE

Each job could be decomposed to the following stages (simulation steps):

- Reading data about the positions of a financial portfolio from the database;
- Executing calculations over the data to perform analyses;
- Storing results from the simulation into the database;

The time of each job's stage is known in advance, according to the performed analyses.

[1]Ivaylo P. Penev is with the Faculty of Computer Science and Engineering, Technical University of Varna, 9000 Varna, Bulgaria, E-mail: ivailopenev@yahoo.com

The stages of reading and storing data from/into the database take time, which is significantly less than the time, necessary for performing calculations.

These observations give reason to apply a strategy, which introduces shifting of the jobs on the time axis. The aim is avoiding the recovering of the stages of different jobs, accessing the common resource simultaneously.

A. Formal description

The strategy is formalized by the help of theory of sets [1].

A set of simulation jobs is defined:

$$JOBS = \{job_1, job_2, ..., job_n\} \tag{1}$$

Each job has time for reading data τ_{READ}, time for calculations $\tau_{EXECUTE}$ and time for storing data τ_{STORE}. The times τ_{READ}, $\tau_{EXECUTE}$, τ_{STORE} are known in advance.

The execution of the whole process, including all the simulation jobs along the time axis is decomposed to a set of time intervals:

$$TIMES = \{t_1, t_2, ..., t_k\} \tag{2}$$

The stages, through which the execution of a job passes through, define a set of possible states, taken by the job in a time interval:

$$STATES = \{\mathrm{Re}ad, Execute, Store\} \tag{3}$$

The process of the execution of all jobs is described by a relation of the Cartesian product of the upper sets:

$Entire\,\mathrm{Pr}ocess = \{\langle t_i, j_j, s_l\rangle\}$, where:

$t_i \in TIMES, j_j \in JOBS, s_l \in STATES$.

When the jobs are performed in series, the total execution time of the whole process is given as a sum of the execution times of each job's stages:

$$t_{\sum} = \sum_{i=1}^{k} t_i$$

In the case of parallel performance the total time is theoretically equal to the maximum execution time of a parallel job:

$t_{\sum} = \max(\sum t_k)$, where t_k are all time intervals, during which a job from the set $JOBS$ is performed.

The practical increase of $t_{\sum}$ is caused by those time intervals, in which different jobs are set at $\mathrm{Re}ad$ or $Store$ state at the same time. These intervals form a subset of the $Entire\,\mathrm{Pr}ocess$ set.

$$ConflictJobs \subset Entire\,\mathrm{Pr}ocess$$

$$ConflictJobs = \left\{\left\langle t_i, j_j, s_l\right\rangle, \left\langle t_i, j_k, s_l\right\rangle\right\} \tag{4}$$

, where

$$j \neq k \wedge (s_l = \mathrm{Re}ad \vee s_l = Store))$$

t_i - A time interval of the whole simulation process execution ($t_i \in TIMES$)

j_j, j_k - Jobs, executed in the t_i interval ($j_j, j_k \in JOBS$)

s_l - State of the jobs j_j, j_k in the interval, which is $\mathrm{Re}ad$ or $Store$ ($s_l \in STATES$)

The total time of the simulation process for the case of parallel execution is in inverse proportion to the count of the elements of the (4) set, i.e. to the set power:

$t_{\sum} \to f(|ConflictJobs|)$, where $|ConflictJobs|$ - power of the set.

The elements of the (4) set present those time intervals, in which different jobs are situated at the same state of access to the common resource at the current time interval. The following strategy is applied to decrease the number of these elements.

A complementary state $WAIT$ is added to the $STATES$ set:

$$STATES = \{Wait, \mathrm{Re}ad, Execute, Store\}$$

This state is added as an initial state at the beginning of each job, accessing the common resource. The job remains in this state until the other jobs occupy the $\mathrm{Re}ad$ state. After the common resource is released, the waiting job moves to the $\mathrm{Re}ad$ state. The job start is shifted along the time axis until the release of the data source.

As a result the $ConflictJobs$ set is shrunk in contrast to the $Entire\,\mathrm{Pr}ocess$ set, which is expanded with new elements. The results derive from the simulation process analysis, which shows that the concurrent access to the common resource is the strongest limiting condition over the efficiency of the whole simulation process.

B. Algorithm for realization of the proposed strategy

The strategy is realized using the following iterative algorithm for the execution of a simulation process, consisted of n parallel jobs:

1. Reading the execution times of the stages of each job.
2. Simulating a scenario with shifting the current job $i, 1 \leq i \leq n$:

2.1. Reading the times $\tau_{READ}, \tau_{EXECUTE}, \tau_{STORE}$ for job $k, 1 \le k \le n, k \ne i$.

2.2. Introducing delay time $\tau_{iWAIT} = \tau_{kREAD}$ for the job i.

2.3. Counting the intervals, in which the jobs access the common resource concurrently after delaying the job i.

2.4. Reading times for the next job k, transition to 2.1.

3. Simulating a scenario with shifting the next job i, transition to 2.
4. Final evaluation – estimating the scenario with minimum recovering of the common resource access.

IV. PROGRAMMING ENVIRONMENT, REALIZING THE PROPOSED STRATEGY

The described strategy is presented by a programming environment for managing the execution of parallel jobs, accessing common data source.

The environment has two main parts:

- Visualizing the parallel execution of jobs, the recovering of states, the scenarios with shifting of jobs;
- Creating a description for each job for starting in the distributed computing environment (submit description file);

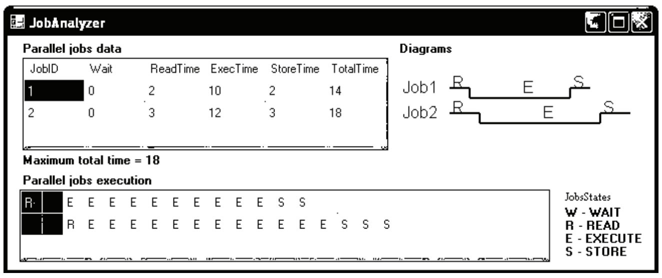
A. Part for visualization of the parallel jobs

This part performs the following main functions:

- Showing the times for each state of all jobs;
- Visualizing the parallel execution of the jobs by the help of diagrams;
- Showing the recovering of states of parallel jobs during the process of execution;

The times for the states of each job, participating in the simulation process, are defined in a data base. The parallel execution is visualized by diagrams, showing the development of the jobs by states. Furthermore, the whole simulation process time is divided to intervals. The state of each job in a time interval is specified, according to the known times for each job's states. The time intervals, in which different jobs occupy the data source simultaneously, are marked. These intervals are the limiting condition over the parallel execution efficiency (the elements of the set (4) from the formal description of the strategy).

By the help of the proposed interface scenarios with shifting of jobs are visualized. At the initial state the parallel jobs recover at the state of reading data from the data base (Fig. 1).



JobAnalyzer

Parallel jobs data

JobID	Wait	ReadTime	ExecTime	StoreTime	TotalTime
1	0	2	10	2	14
2	0	3	12	3	18

Maximum total time = 18

Parallel jobs execution

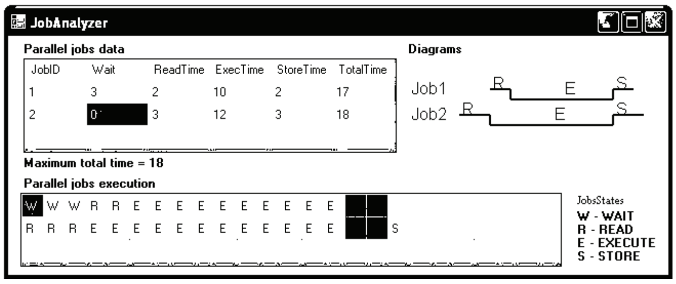
```
R ■ E E E E E E E E E E S S
■ ■ R E E E E E E E E E E E E S S S
```

JobsStates
W - WAIT
R - READ
E - EXECUTE
S - STORE

Fig. 1. Position of the jobs at the initial state

The described algorithm is applied. Scenarios with time shifts for each job are generated. The example presents managing of two parallel jobs.

The first job is shifted with two time intervals, until the second one finishes reading data and releases the common resource. The recovering of states shows, that after shifting of the first job, the parallel jobs become concurrent to the data source at the states of storing data. This means that the scenario would be ineffective for the whole simulation process and such ordering of jobs is improper (Fig. 2).



JobAnalyzer

Parallel jobs data

JobID	Wait	ReadTime	ExecTime	StoreTime	TotalTime
1	3	2	10	2	17
2	0	3	12	3	18

Maximum total time = 18

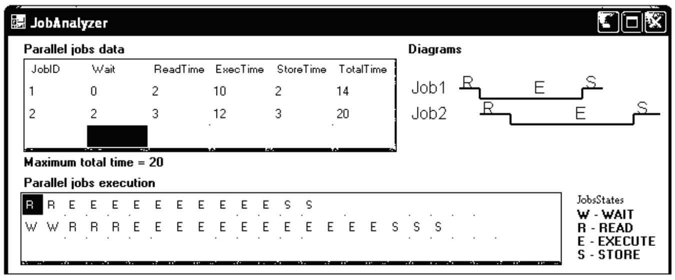
Parallel jobs execution

```
W W W R R E E E E E E E E E E ■ ■
R R R E E E E E E E E E E E E ■ ■ S
```

JobsStates
W - WAIT
R - READ
E - EXECUTE
S - STORE

Fig. 2. Variant of job shifting

In the next scenario the second job is shifted with two intervals, until the first one ends reading data and releases the resource. The recovering of states shows, that the shift of the second job in relation to the first one prevents the concurrent access to the common data source. Although the maximal total time is greater than the same time in the previous scenario, this case is expected to avoid the limiting condition over the simulation efficiency. This should be the order of starting of the parallel execution in the distributed computing environment (Fig. 3).


JobAnalyzer

Parallel jobs data

JobID	Wait	ReadTime	ExecTime	StoreTime	TotalTime
1	0	2	10	2	14
2	2	3	12	3	20

Maximum total time = 20

Parallel jobs execution

```
R R E E E E E E E E E E S S
W W R R R E E E E E E E E E E E E S S S
```

JobsStates
W - WAIT
R - READ
E - EXECUTE
S - STORE

Fig. 3. Next variant of job shifting

B. Creating a description for each job for starting in the distributed computing environment (submit description file)

Each job must have a description to be started in the distributed environment. The description is stored in a file, called submit description file, which is created by the presented programming environment.

The file matches a computer from the pool with a portfolio from the database. The computer, which host name is specified in the description file, calculates the portfolio and stores the results into the database.

The description file has specified format, which includes information about the simulation execution and the work of the pool:

- Name of the executable, sent to the computers in the environment;
- Arguments – name of the portfolio to be calculated in background mode;
- Requirements – host name of the computer from the pool to calculate the portfolio;

V. CONCLUSIONS AND FUTURE WORK

Up to the current moment the presented strategy is in a process of development. The environment simulates small number of jobs. The purpose is building an effective model for researching the real execution of many parallel jobs with concurrent access to the common resource.

The formal description of the strategy will be complicated. After a job finishes, the computing resource is free for next job. Further analysis about the state of the other jobs is required. Complementary criteria will be defined, estimating the current results from the strategy appliance.

ACKNOWLEDGEMENT

The work presented in this paper was partially supported within the project BG 051PO001-3.3.04/13 of the HR Development OP of the European Social Fund 2007-2013.

REFERENCES

[1] L. Lovasz, K. Vesztergombi, “Discrete Mathematics”, Lecture Notes, Yale University, 1999.

[2] I. Penev, A. Antonov, "Realization of Portfolio Management System in a Distributed Computing Environment", International Scientific Conference Computer Science, Conference Proceedings, under print, Sofia, Bulgaria, 2009.

[3] http://cs.wisc.edu/condor/.