Journal of Applied Sciences

ISSN 1812-5654

Journal of Applied Sciences 11 (6): 932-942, 2011
ISSN 1812-5654 / DOI: 10.3923/jas.2011.932.942
© 2011 Asian Network for Scientific Information

Improving the Design of Parallel Applications using Statistical Methods

[1]C.M. Amarandei, [2]D. Lepadatu and [3]S. Caraiman
[1]Depatment of Computer Science and Engineering, Gheorghe Asachi Technical University, 27 Mangeron Blvd, Iasi, 700050, Romania
[2]Department of Civil Engineering, Gh. Asachi Technical University, Iasi, 43 Mangeron Blvd, Iasi, 700050, Romania
[3]Department of Computer Science and Engineering, Gheorghe Asachi Technical University, 27 Mangeron Blvd, Iasi, 700050, Romania

Abstract: The aim of this study is to analyze data produced by a parallel application using statistical investigation methods like design of experiments to determine faults in the design or implementation of the application. Using this technique we develop a model to verify the design of a parallel application by determining the factors influencing its performance and describe the behavior of the application based on these factors. Experimental studies were performed on a parallel rendering application based on a sort-last domain decomposition technique. The following characteristics of the parallel application were analyzed: processing time, communication time and size of the data exchanged between processes. The performed analysis revealed the limitations of the application design in various runtime conditions offering strong clues on how certain stages of the application should be redesigned.

Key words: Design of experiment, parallel application design, domain decomposition model

INTRODUCTION

According to Foster (1995), the design of a parallel application involves four stages: data partitioning, communication design, agglomeration and mapping. Even if the verifications required by Foster are followed, the entire solution chosen in a certain stage can prove to not have been the most appropriate. This can have a major impact on the performance of the parallel application. Following this model a bad application design can result in any of the 4 stages. This could not necessarily be due to the supplementary verifications required, but due to run-time conditions, which also need to be taken into consideration when a prediction for the performance of the parallel application is required.

During the design stage the size of the datasets used in the actual application runs is not usually known. Following the design stages and rules proposed by Foster (1995) the data is separated in distinct blocks that can be individually processed. After establishing the data-partitioning scheme, it is necessary to define the communication patterns, which can be a challenging task. If errors emerge in any of the partitioning or communication design stages, these can be propagated in the agglomeration and mapping stages.

This rationale emphasizes analytic modeling of performance. Yet parallel programming is first and foremost an experimental discipline. Cost effective approaches to parallel program design can rarely be accomplished based entirely on theory because of the flexibility and ease of modification of software on one hand and the complexity of parallel computer systems on the other (Foster, 1995).

Due to a large number of design strategies and possibly influential factors, there are numerous aspects that need to be clarified by the experimental studies:

- The factors that influence a certain performance metric
- Whether there is any interaction among factors
- The quantification of the effect of each factor and of the interactions between them
- Factor settings providing optimal performance
- Limitations enforced by these settings on the application

One approach would be to use heuristic methods, but in this case the behavior of an application cannot be quantified. Heuristics are stochastic in nature, meaning that repeated runs of a heuristic produce different

Corresponding Author: Cristian-Mihai Amarandei, Department of Computer Science and Engineering, Gheorghe Asachi Technical University, 27 Mangeron Blvd, Iasi, 700050, Romania
Tel: +40-232-268783/1304 Fax: +40-232-231343

behaviors. Their approximate nature means that while they should produce good solutions relatively quickly, they will actually produce improved solutions over extended run times (Ridge and Kudenko, 2007).

In this study, we propose a novel model for designing parallel application based on performance analyses, leading to objective conclusions valid over the possible design faults. Our approach is to use a systematic statistical design of experiments strategy to facilitate the design task. Thus, after implementing an application prototype, its evolution can be easily followed and possible design errors can be identified. The advantages of using a statistical DoE (Design of Experiments) approach are significant: a proper DoE should allow the application designer to obtain the maximum information with the minimum of experiments (Totaro and Perkins, 2005).

BACKGROUND AND RELATED WORK

Statistical DoE offers a way for the researcher to determine, before the runs are made, which specific configurations to simulate so that the desired information can be obtained with the least amount of Law and Kelton (2000). Additionally, an experimental design that has been properly executed and analyzed: (1) facilitates the identification of effects of various factors (variables) that might affect performance and (2) helps to determine if a particular factor has a significant effect or if the observed difference is due to random variations that resulted from errors in measurement and uncontrolled parameters (Jain, 1991).

The DoE technique can be successfully applied to black-box cases when the aim is to optimize a quality characteristic (system response) by adjusting input data (factors). Input data are known as factors/variables, which can be known or controlled. There can be other factors that influence the system but without the possibility of controlling them. Thus they represent a source of uncertainty and they are called unknown or uncontrolled factors. The output of the system is called a response variable and is actually a quality feature that has to be studied and optimized. More in-depth information about statistical DoE and empirical model building can be found by Jain (1991), Box *et al.* (1978), Kleijnen *et al.* (2004) and Montgomery (2001). Several guidelines for conducting empirical research in software engineering are presented (Kitchenham *et al.*, 2002).

As shown by Ruthruff *et al.* (2006) the use of an experimental program analysis paradigm can help researchers and software developers identify limitations of analysis techniques, improve existing experimental program analysis techniques and create new experimental program analysis techniques. These techniques might be able to draw inferences about the properties of software systems in cases in which more traditional analyses have not succeeded (Ruthruff *et al.*, 2006). For example, rigorous statistical techniques are used for embedded Java performance analysis (Rao and Murakami, 2008). Researchers build regression models that relate overall Java system performance to various micro-architecture metrics and their interactions. The models developed are easy to interpret and achieve accuracies that are close to measurement errors. Through this modeling technology, the system architect is advised to simultaneously tune the Java runtime parameters for optimal performance.

The use of efficient experimental designs within a sequential strategy also seems to be a good policy for selecting the components of an application that offer optimum results for a certain problem. Such an approach is described by Stewardson *et al.* (2002) for selecting just a few of the many possible combinations of Genetic Algorithms for use in complex scheduling problems. As the selection must be balanced and cover each type of parameter and combinations of these, equally, the researchers use an experimental design as such a method can reduce the level of uncertainty and quickly cut through complexity. The DoE techniques are also used (Ridge and Kudenko, 2007) where a predictive model of the performance of combinatorial optimization heuristic over a range of heuristic tuning parameter settings and problem instance characteristics was built.

Various statistical approaches for analyzing the performance of parallel applications have been proposed recently by Zheng *et al.* (2010), Barnes *et al.* (2008) and Whiting *et al.* (2004). Nevertheless, neither of them addresses the use of such techniques for building a generic parallel application design model. For example, Zheng *et al.* (2010) used statistical models to predict the performance of large scale parallel applications using the performance of sequential execution blocks. This is done by scaling down the simulation of the problem size to capture the characteristic of the application. Using machine learning techniques, the knowledge gained is then turned into a statistical model for each sequential execution block, which can be used to predict the application at full scale running on an emulator. Barnes *et al.* (2008) used multivariate regression to predict the performance on large processor configurations using training data obtained from smaller numbers of processors. Even though this method can be used to predict parallel program scalability, the technique is unable to offer any information about the factors influencing the performance of the parallel application.

The statistical experimental design techniques are used by Whiting *et al.* (2004) to evaluate performance of the parallel implementation of a complex computational phylogenomics problem. Researchers also present several designs of experiment procedures like full factorial, fractional factorial and D-optimal design. Evaluation of the parameters is performed using a D-optimal design procedure and several conclusions are drawn regarding the influence of search space, chosen ratchet algorithm and number of processors on the performance of the application.

In contrast with other study, our approach represents a design strategy for parallel applications based on a predictive performance model. The aim of the statistical experimentation stage is twofold: to determine the factors influencing the performance of the parallel applications and to predict the application performance in various execution conditions during the design stage.

PROPOSED PARALLEL APPLICATION DESIGN MODEL

The model proposed for the design of a parallel application requires 3 stages presented in Fig. 1. The first stage contains the classic steps of parallel application design as presented by Foster. The result of this stage represents the prototype of the parallel application.

After a design is complete, an important role is played by performance models (Foster, 1995). Valuable information about both an application design and its implementation can be acquired by comparing the observed and predicted execution times. These values will seldom completely agree due to the idealized nature of the models used in design stage. If major discrepancies emerge, these are due either to an incorrect model or to an inadequate implementation. In the case of an incorrect model, empirical results can be used to determine the deficiencies of the model and to reassess the quantitative tradeoffs used to justify design decisions. In the second case, we can use the model to identify areas in which the implementation can be improved.

The second stage of the proposed design model aims at evaluating the performance of the application prototype and requires the definition of an experimental design plan, performing the experiments, statistical analysis and interpretation of the results. In this step functionality tests and performance measurements are performed on the prototype: processing time, communication time, size of data transferred between processors, etc.

The statistical analysis can provide information about the application performances and can help in eliminating the factors that don't have a direct influence over the response. Moreover, the behavior of the parallel application can be modeled based on the resulting data set. Thus, as illustrated in the following section, a prediction can be made for the response variables as a function of the input factors.

Following the statistical analysis, if the obtained information is not satisfactory it can be decided to reconsider one or more application design stages: for

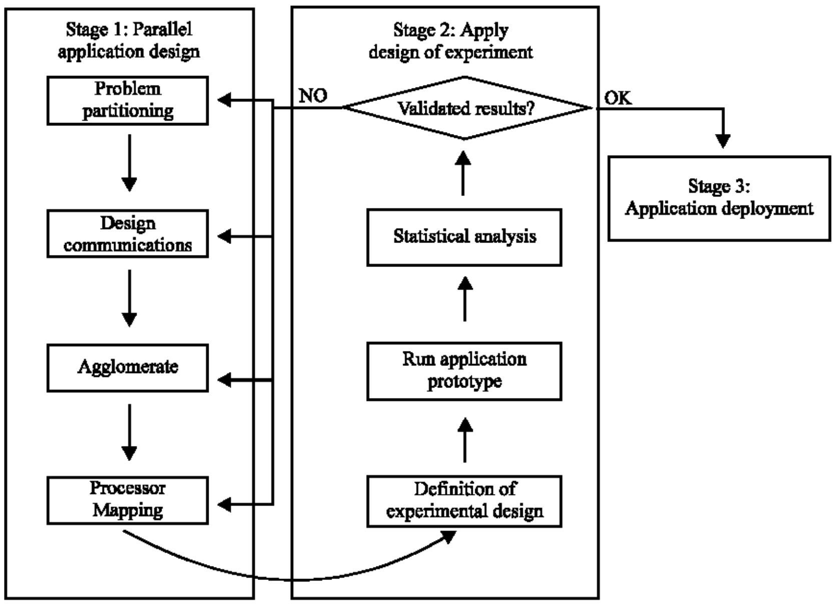

Stage 1: Parallel application design
Problem partitioning
Design communications
Agglomerate
Processor Mapping
Stage 2: Apply design of experiment
NO
Validated results?
OK
Statistical analysis
Run application prototype
Definition of experimental design
Stage 3: Application deployment

Fig. 1: DoE based parallel application design

example, the communication patterns are good, but there is a bad design of the data-partitioning model.

The first two stages are reiterated until the experimental results are validated against the imposed performance metrics. At this point the third and final stage can be employed: application deployment.

PARALLEL APPLICATION ANALYSIS USING DOE TECHNIQUES

The performance of a parallel application is usually evaluated against factors such as speedup, efficiency, scalability that assume measurements of processing time, communication time and idle time. Using statistical analysis of experimental data by way of analysis of variance (ANOVA) techniques, we are able to identify main effects and interaction of factors that explain such performance metrics. The ANOVA (analysis of variance) table for a response variable is a useful tool for identifying main and interaction effects of factors that are statistically significant.

The ANOVA table has six columns: the source of the variability, the sum of squares (SS) due to each source (the variation), the Degrees of Freedom (df) associated with each source, the Mean Squares (MS), which is the ratio SS/df (the variance), the F-statistics, which are the ratios MS/MSError and the P-values for the F statistics. A large ratio of the mean squares (the F-statistic) implies that the amount of variation explained by the considered source is large in comparison with the residual error. The p-value is of particular interest to us, since it serves as an indicator of “statistical significance" as explained in the next section.

The steps that need to be followed for preparing an experiment are:

- **Defining the objectives of the experiment:** Our underlying goal of this work is to demonstrate the effectiveness of a statistical DoE strategy when evaluating the performance of parallel applications and identify the design faults. To this end, the specific objectives of our experiments are to quantify the effects of possible influential factors on the performance of the parallel application. Using these effects, we then develop empirical models, which can be used to predict performance of the parallel application running over the range of values examined in this study
- **Selecting the input factors (and their levels):** The next step in the experimental design is the selection of potential factors that may impact the performance of a parallel application
- **Selecting the response variables:** Next we must select the performance responses of the parallel application
- **Selecting the appropriate design:** We use a two level full factorial design. For simplicity and computational purposes, it is often useful to code the factor levels as a + /- or 0/1 level. Factorial designs are efficient (instead of conducting a series of independent studies, we are effectively able to combine these studies into one) and are the only effective way to examine interaction effects
- **Simulation and data collection:** In this step the parallel application tests begin. Depending on the number of input factors, the applications must run n times and at each run we must use input factors values according to the design plan and collect the response
- **Computing the main and interactive effects:** Intuitively, the main effect of a factor refers to the average change in a response variable produced by a change in the level of the factor. The interaction effects are those combinational effects that two factors have on the two response metrics (Totaro and Perkins, 2005). An interaction effect means that two factors interact if the performance response due to factor i at level m depends on the level of factor j. In other words, the relative change in the performance response due to varying factor i is dependent on the level of factor j. Having examined the apparent main effects of each of the factors on the response metrics, we next turn our attention to interaction effects, which are those combinational effects that two factors have on the two response metrics. Thus, two-way factor interaction effects plots can be used to visualize the performance changes that result from the combined varying of two factors from their-levels to their + level
- **Building an empirical model for the selected response variables:** After running the application using the selected input factors, their influence on the response variables is analyzed. We are now ready to build empirical models that are based on the data we have collected. The statistical DoE approach is what facilitates this. The basic steps of the model-building process are: model selection, model fitting and model validation. These three basic steps are used iteratively until an appropriate model for the data has been developed. In the model selection step, the main and interactive effects determined from the ANOVA table and assumptions about the process are used to determine the form of the model to be fit to the data. Then, using the selected model and

possibly information about the data, an appropriate model-fitting method is used to estimate the unknown parameters in the model. When the parameter estimates have been made, the model is then carefully assessed to see if the underlying assumptions of the analysis appear plausible. If the assumptions seem valid, the model can be used to answer the scientific or engineering questions that prompted the modeling effort. If the model validation identifies problems with the current model, however, then the modeling process is repeated using information from the model validation step to select and/or fit an improved model

- **Interpretation of the results:** This final stage aims at correlating the effects of the considered input factors on the analyzed response of the parallel application and should be used to validate/invalidate the design of the application according to the imposed performance objectives

CASE STUDY: SORT-LAST PARALLEL RENDERING

In order to verify the proposed application design model, the following steps were followed:

- Creating a parallel application that uses domain decomposition
- Definition of a full factorial DoE plan in order to analyze performances of the parallel application
- Application run and data collection
- Analysis of factors variation and regression
- Interpretation of the results in order to find possible design faults

Parallel application: We demonstrate the use of the proposed model in analyzing the design of a parallel rendering application. The rendering process consists of two main stages: geometry processing (model-view transformations, lighting) and rasterization (shading, visibility processing). Assigning to each processor a subset of the graphic primitives in the scene usually parallelizes geometry processing. Assigning a portion of pixel computations to each processor parallelizes Rasterization.

In essence, computing the effect of each geometric primitive on each pixel represents the rendering task. Due to the arbitrary nature of the model-view transformations, a graphic primitive can fall anywhere on (or off) the screen. Thus, rendering can be seen as a problem of sorting the graphic primitives to the screen (Molnar *et al.*, 2008). The stage at which this sort takes place determines the structure of the resulting parallel rendering system. Generally, this sort can take place anywhere in the rendering process: during geometry processing (sort-first) between geometry processing and rasterization (sort-middle) or during rasterization (sort-last). In sort-first rendering raw primitives are redistributed before determining their screen-space parameters. Sort-middle rendering redistributes screen-space primitives. Sort-last rendering redistributes pixels, samples, or pixel fragments.

In sort-last parallel rendering, also known as object-space parallel rendering, each processing node is responsible of rendering a block of data, irrespective of their visibility on the screen (Fig. 2). Each working node is assigned a block of data and renders the corresponding image. In the last stage, the master process gathers the images from all workers and composes the final image by depth-sorting the pixels.

Our analysis involved such a sort-last parallel rendering application designed using the message passing programming model. The application is developed in C++ and parallelization has been achieved through the use of LAM 7.1.2/7.1.4 implementation of the MPI-2 standard. Even though rendering is usually a callback process, in this case we are interested in analyzing the behavior of the application when producing very large images of very large datasets. The input datasets used represent point clouds acquired with a 3D laser scanning device with position (x, y, c) and color (r, g, b, a) information associated to each point.

Designing the experiment: Since our goal in this paper is to illustrate the effectiveness of the statistical DoE strategy, we select only a subset of three factors that can be considered for the parallel rendering application: X_1-size of input data, i.e., number of points in point cloud (min value-100000, max value-4000000), X_2-final image size, i.e., resolution in pixels (min value-480000, max value-50000000) and X_3-number of processors (min value-6, max value-8).

We consider the following performance responses for the parallel application: Y_1-the amount of data exchanged between processes, Y_2-processing time and Y_3-communication time.

We use a two level full factorial design. Table 1 shows all possible combinations of factor levels where each combination corresponds to a simulation scenario.

The environment used to test the proposed model consists in a cluster with the following configuration:

- One front-end computer with 4×3.66 GHz Intel Xeon processors, 4×146 GB hard drive and 8 GB of RAM
- 12 computing nodes with 1×2.33 GHz Intel Core2 Duo

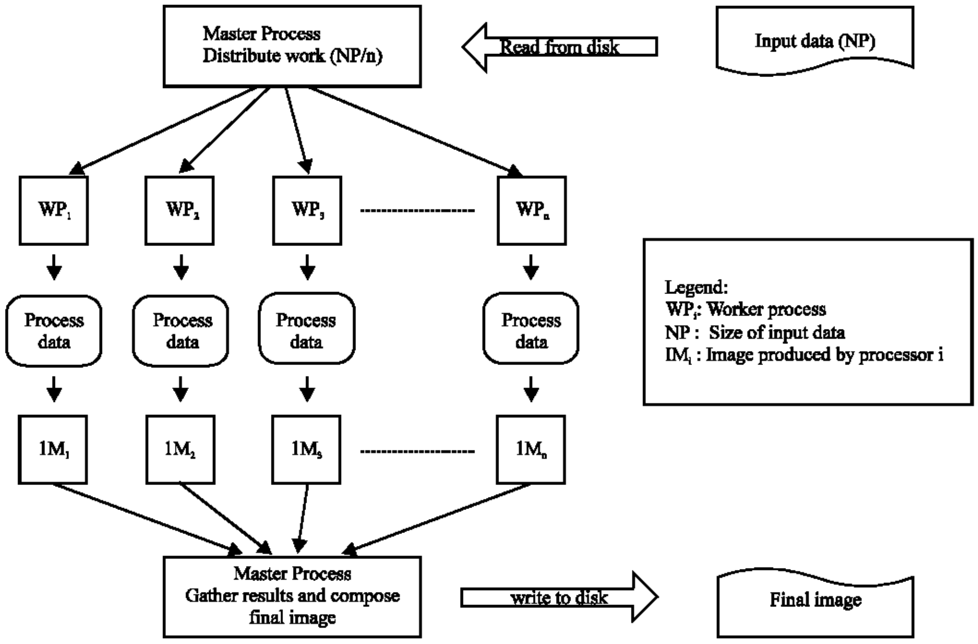


Fig. 2: Sort-Last parallel rendering

CPU, 1×160 GB hard drive and 2GB of RAM with Gigabit Ethernet card connected with CAT6 cables via a Gigabit switch

Table 1 presents the encoded values used for performing the simulations and the values of the response variables acquired according to the design plan. The ANOVA tables for the selected response variables are presented in Table 2.

The analysis of variance can be used to test hypotheses about the main factor effects of X_1, X_2, X_3 and their interactions ($X_1 \times X_2$, $X_1 \times X_3$ and $X_2 \times X_3$) on the response. In this example there are six hypotheses to be tested for each response: main effect $X_1/\ X_2/X_3$ is not significant for the variation of the response, interaction effect $X_1 \times X_2/\ X_1 \times X_3/\ X_2 \times X_3$ is not present. For example, for main effect X_1, the null hypothesis means that there is no significant variation of the response whether the input data set contains 100000 or 4000000 points. For an interaction effect, the null hypothesis means that the respective two main effects are independent.

One way to report the results of a hypothesis test is to state that the result of the null hypothesis was or was not rejected at a specified α-value or level of significance. The p-value (Table 2) is the smallest level of significance that would lead to rejection of the null hypothesis with the given data. If any p-value is near zero, this casts doubt on the associated null hypothesis, that is, there is a main effect due to the considered factor. In order to determine whether a result is statistically significant a bound for the p-value needs to be chosen. It is common to declare a result significant if the p-value is less than 0.05 or 0.01 (Montgomery, 2001).

Using the experimental data we now try to model the response variables as functions of the input factors. For all three responses we first develop a linear regression model that defines the functional relationship between the influential factors and the performance metrics. Other regression models can be used if the linear regression doesn't provide an adequate model. The general form of the function describing the behavior of a response Ycan be written as:

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_j x_j + \sum_{i<j}^{p} \beta_{ij} x_i x_j + \sum_{i<j<k}^{p} \beta_{ijk} x_i x_j x_k + \ldots + \varepsilon \qquad (1)$$

where, p represents the number of variables (factors) and ε represents the residual error (the difference between the predicted result and the one measured experimentally). Detailed information about regression techniques can be found (Montgomery and Runger, 2007).

In the following steps we estimate the model parameters for each response and check the adequacy of the model in order to validate it.

Analysis of response Y_1-amount of data exchanged between processes: The results obtained when running the application on 6 and 8 processors reveals the fact that response Y_1 depends solely on factor X_2 (final image size),

Table 1: Encoded design plan and measured values for the response variables

Runs	X_1	X_2	X_3	Y_1	Y_2	Y_3
1	0	0	0	7500	2.40813	0.661006
2	1	0	0	7500	20.8237	2.288
3	0	1	0	781250	177.86	65.7456
4	1	1	0	781250	199.783	66.1748
5	0	0	1	7500	3.02641	0.880815
6	1	0	1	7500	21.7683	2.55107
7	0	1	1	781250	243.713	89.9857
8	1	1	1	781250	264.943	90.3144

Table 2: Analysis of variance - ANOVA table for responses Y_1, Y_2 and Y_3

Responce	Source	SS	df	MS	F	P-value
Y_1	X_1	0.0005	1	0.0005	1	0.5
	X_2	1.19738E+12	1	1.19738E+12	2.45223E+15	1.34E-8
	X_3	-0.0005	1	-0.0005	-1	1
	X_1X_2	-0.0005	1	-0.0005	-1	1
	X_1X_3	-0.0005	1	-0.0005	-1	1
	X_2X_3	-0.0005	1	-0.0005	-1	1
	Error	0.0005	1	0.0005		
	Total SS	1.19738E+12	7			
Y_2	X_1	806.2	1	806.2	6207.59	0.0081
	X_2	87837.6	1	87837.6	676315.45	0.0008
	X_3	2197	1	2197	16916.4	0.0049
	X_1X_2	4.5	1	4.5	34.6	0.1072
	X_1X_3	0.0168	1	0.0168	0.13	0.7802
	X_2X_3	2094.7	1	2094.7	16128.12	0.005
	Error	0.1298	1	0.1298		
	Total SS	92940.2	7			
Y_3	X_1	2.1	1	2.1	795.67	0.0226
	X_2	11692.2	1	11692.2	4525903.49	0.0003
	X_3	298.4	1	298.4	115523.57	0.0019
	X_1X_2	0.8	1	0.8	312.01	0.036
	X_1X_3	0.00004	1	0.00004	0.16	0.7588
	X_2X_3	286.8	1	286.8	111002.1	0.0019
	Error	0.0026	1	0.0026		
	Total SS	12280.3	7			

while factors X_1 and X_3 have no influence on this response. The ANOVA (Table 2) confirms this observation: the null hypothesis H_0: $\beta = 0$ is rejected. It is important to notice that this behavior is also consistent with the analysis of parallel rendering methods provided (Molnar *et al*., 2008).

Following this observation we infer that Y_1 is linearly dependent on factor X_2, which can be stated as:

$$Y_1 = \beta_0 + \beta_2 x_2 + \varepsilon \tag{2}$$

where, β_0 and β_2 can be determined using the values provided in Table 1 for input factor X_2 and the corresponding measured response and ε represents the prediction error. The following values for the coefficients are obtained:

$$\begin{cases} \beta_0 = 0 \\ \beta_2 = 0.15625 \end{cases} \tag{3}$$

providing that the equation describing the behavior of Y_1 has the following form:

$$Y_1 = 0.15625 \cdot x_2 \tag{4}$$

where, x_2 represents the resolution of the resulting image.

Table 3 shows that there is no difference between the measured evolution of response Y_1 with respect to X_2 and the values computed using Eq. 4, i.e., $\epsilon = 0$. Factor X_1 (size of input point cloud) doesn't affect the size of communicated data, but it influences the processing time as it results from the following analysis.

Analysis of response Y_2-processing time: We can see from Table 2 that the largest effects on response Y_2 are X_1, X_2, X_3 and the X_1X_2 interaction. Choosing a value of 0.01 for the smallest level of significance that would lead to rejection of the null hypothesis with the given data, the regression model used to obtain the predicted values is:

$$Y_2 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{23} x_2 x_3 + \varepsilon \tag{5}$$

Using the values in Table 1 the aim is to compute the regression coefficients, β, for the Eq. 1 where we consider the response to be Y_2. These coefficients can be computed by minimizing the residual error using linear regression (Barnes *et al.*, 2008). Using the values obtained for the regression coefficients Eq. 5 becomes:

$$Y_2 = -6.52*10^{-1} + 4.92*10^{-6}x_1 - 3.75*10^{-7}x_2 + 1.25*10^{-1}x_3 + 6.54*10^{-7}x_2x_3 + \varepsilon \quad (6)$$

The measured and predicted values for response Y_2 are presented in Table 4. We validate the regression model in Eq. 6 used for response Y_2 using standard methods for residual analysis (normal probability plot of residuals and residuals standardization) as no anomalies in the fitted model were revealed (Fig. 3a).

Using the function that describes response Y_2, the behavior of the system can be predicted in the case of varying the number of processors as in Table 5. Thus, using the β coefficients determined for 6-8 processors, the processing time for 10-12 processors can be computed. These values determined with Eq. 6 for Y_2 are presented in Table 4 as they are compared with the actually measured values.

Analysis of response Y_3-communication time: In order to analyze the communication time, the same rationale as for Y_2 can be used. It can be observed from the Anova table (Table 2) that the response Y_3 depends on factors X_2 and X_3 and the X_2 X_3 interaction. In this case, the following equation can be used to model the response:

$$Y_3 = \beta_0 + \beta_2 x_2 + \beta_3 x_3 + \beta_{23} x_2 x_3 + \varepsilon \quad (7)$$

Using the β coefficients determined for Y_3 Eq. 7 becomes:

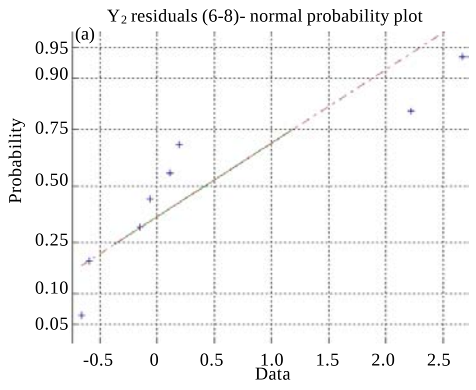
$$Y_3 = -1.04*10^{-1} - 1.35*10^{-7}x_2 + 1.22*10^{-2}x_3 + 2.42*10^{-7}x_2x_3 + \varepsilon \quad (8)$$

Table 3: Evolution of measured and computed response Y_1 with respect to input factor X_1

Image resolution (pixels)	Y_1-Size of communicated data (KB)-measured-	Y_1-Size of communicated data (KB)-computed with Eq. 5-
480000	7500	7500
1000000	15625	15625
10000000	156250	156250
25000000	390625	390625
50000000	781250	781250
100000000	1562500	1562500
120000000	1875000	1875000
150000000	2343750	2343750
160000000	2500000	2500000
170000000	2656250	2656250
172000000	2687500	2687500
176000000	2750000	2750000
178000000	2781250	2781250
179000000	2796875	2796875

The regression model in Eq. 8 is validated by observing a normal distribution of the residuals, just like in the case of response Y_2 (Fig. 3b). Next, they are used to calculate the prediction for the communication time in the case of using 10-12 processors. Table 6 shows the measured and predicted values for Y_3 in both cases: 6-8 and 10-12 processors.

Interpretation of the results: The design of parallel rendering algorithms can be a difficult task. In some cases existing sequential algorithms can be easily transformed in parallel algorithms, while in other cases new algorithms need to be designed from scratch. Regardless of their origin, most parallel algorithms introduce overheads not present in their sequential counterparts. These overheads can appear due to: inter-processor communication, load imbalances, additional or redundant computations, increased storing demands for auxiliary or replicated data structures. Some of these concepts are specific to parallel algorithms while others (data coherence, object-space to image-space data mapping) correspond to the rendering problem.



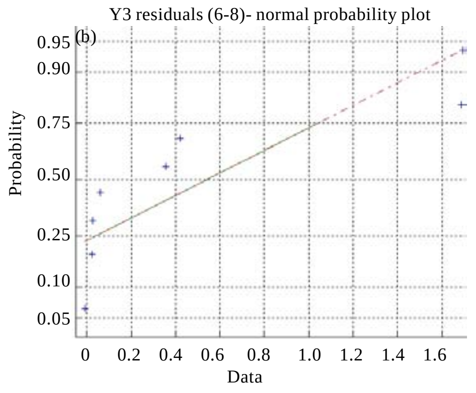


Fig. 3: Normal probability plot of the residuals: (a) for response Y_2 and (b) for response Y_3

Table 4: Measured and predicted values for processing time (response Y_2) in the case of 6-8 processors (left) and 10-12 processors (right)

6-8 processors			10-12 processors		
Y_2 measured	Y_2 predicted	ε	Y_2 measured	Y_2 predicted	ε
2.40813	2.29	0.11	3.62407	4.05	-0.43
20.8237	21.49	-0.66	22.1726	23.24	-1.07
177.86	177.92	-0.06	1111.68	1103.87	7.81
199.783	197.12	2.67	1131.72	1123.06	8.66
3.02641	3.17	-0.15	4.27239	4.93	-0.66
21.7683	22.36	-0.59	23.2734	24.12	-0.85
243.713	243.53	0.19	1359.53	1338.08	21.45
264.943	262.72	2.22	1374.250	1357.27	16.98

Table 5: Values of input factors used for predicting response Y_2 in the case of varying the number of processors

X_1	X_2	X_3
100000	480000	10
4000000	480000	10
100000	1.79E+08	10
4000000	1.79E+08	10
100000	480000	12
4000000	480000	12
100000	1.79E+08	12
4000000	1.79E+08	12

Table 6. Measured and predicted values for communication time (response Y_3) in the case of 6-8 processors (left) and 10-12 processors (right)

6-8 processors			10-12 processors		
Y_3 measured	Y_3 predicted	ε	Y_3 measured	Y_3 predicted	ε
0.661006	0.6004	0.0606	1.11392	1.1134	0.0006
2.288	0.6004	1.6876	2.79507	1.1134	1.6817
65.7456	65.7535	-0.0079	407.274	408.6592	-1.3852
66.1748	65.7535	0.4213	408.065	408.6592	-0.5942
0.880815	0.8569	0.0239	1.34075	1.3698	-0.0291
2.55107	0.8569	1.6942	3.03985	1.3698	1.6700
89.9857	89.9584	0.0273	493.946	495.2499	-1.3039
90.3144	89.9584	0.3560	494.171	495.2499	-1.0789

For the discussed parallel rendering application we aimed at describing the application behavior with the help of an equation and using this equation to study (and predict) the performance of the application when varying the number of processing units.

Using the empirical models built for the three responses, we note that both the processing time and communication time depend on the resolution of the final the local result of each rendering processor to a compositor node. The larger the produced image, the more time it takes to communicate it. Moreover, this can represent a communicational bottleneck in the parallel application. Another important conclusion regards the number of processor used for distributing the rendering task. Using a larger number of rendering processors doesn't seem to provide better total processing times. This scalability issue could be solved by replacing the final image compositing stage with a more efficient mechanism, for example routing the messages in the network using predefined paths in a grid or a binary tree or using a planned image compositing.

Considering the above observations, in the event of simultaneously executing the application by several users or if maximum memory allocation limits are enforced, the application becomes extremely limited and we expect poor performances unless parts of the application are redesigned.

CONCLUSIONS AND FUTURE WORK

In this study we proposed a model for rigorous testing the design of a parallel application by defining an efficient test set and predicting the performances of the application in other running conditions. We have shown that this can be achieved using statistical analysis that allows for an easier identification of the errors appearing in each design stage. The proposed method facilitates the elimination of the factors that do not influence the analyzed response of the application, while giving a better view on the factors that have the greatest impact on the performance of the parallel application.

The interpretation of the results obtained using the statistical analysis has to take into account the imposed objectives. In this study we presented an example where the evolution of communication time, processing time and volume of transferred data were followed in the case of a parallel rendering application. In this example the statistical analysis emphasized that deficiencies exist in the parallel application model. In this case, the bad application design leads to poor scalability and unsatisfactory speedup.

The DoE techniques also allow the designer of a parallel application that uses domain decomposition to test the performance for real input parameters without having to effectively run the application. The estimations obtained by statistical analysis can be used to optimize the resources needed by the application at a certain moment and also to improve the application in the sense of dynamically adapting the execution to the input/output data. If the programming environment allows it, it could also be useful to dynamically modify the number of processes and/or processors used by the parallel application. Using the proposed method can also bring useful insight regarding certain runtime conditions, which can be obtained based on the considered parameters. This is particularly useful for predicting the necessary resources for the parallel application. In this respect, a merely considered but important factor is the fact that the application isn't the only one running on a cluster. Thus, the load of the internal network of the cluster varies due to data transferred by all applications. Moreover, the resources of the cluster can be reserved for several purposes (i.e., academic, research or commercial purposes). An application that is not aware of this will have different performances at different moments of time.

REFERENCES

Barnes, B.J., B. Rountree, D.K. Lowenthal, J. Reeves, B.R. de Supinski and M. Schulz, 2008. A regression-based approach to scalability prediction. Proceedings of the 22nd Annual International Conference on Supercomputing, June 7-12, Island of Kos, Greece, pp: 368-377.

Box, G.E.P., W.G. Hunter and J.S. Hunter, 1978. Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building. John Wiley and Sons, New York, pp: 306-351.

Foster, I., 1995. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Weslley Publishing Co., New York, ISBN-13: 978-0201575941, pp: 430.

Jain, R., 1991. The Art of Computer System Performance and Analysis. John Wiley and Sons, New York.

Kitchenham, B.A., S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam and J. Rosenberg, 2002. Preliminary guidelines for empirical research in software engineering. IEEE Trans. Software Eng., 28: 721-734.

Kleijnen, J.P.C., S.M. Sanchez, T.W. Lucas and T.M. Cioppa, 2004. A users guide to the brave new world of designing simulation experiments. INFORMS J. Comput., 17: 263-289.

Law, A.M. and W.D. Kelton, 2000. Simulation, Modeling and Analysis. McGraw-Hill Higher Education, New York, ISBN: 978-0070592926.

Molnar, S., M. Cox, D. Ellsworth and H. Fuchs, 2008. A sorting classification of parallel rendering. Proceedings of the ACM SIGGRAPH ASIA 2008 Courses, Singapore, Dec. 10-13, ACM, New York, USA., pp: 1-11.

Montgomery, D.C., 2001. Design and Analysis of Experiments. 5th Edn., John Wiley and Sons, New York, ISBN: 047148735X.

Montgomery, D.C. and G.C. Runger, 2007. Applied Statistics and Probability for Engineers. 4th Edn., John Wiley and Sons, New York, ISBN-13: 978-0470067215.

Rao, P. and K. Murakami, 2008. Using statistical models for embedded java performance analysis. Proceedings of the Students Symposium held in Conjunction with the 15th International Conference on High Performance Computing (HiPC'08). http://www.c.csce.kyushu-u.ac.jp/lab-db/papers/paper/pdf/2008/rao08-2.pdf.

Ridge, E. and D. Kudenko, 2007. Analyzing heuristic performance with response surface models: Prediction, optimization and robustness. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, July 07-11, London, England, UK., pp: 150-157.

Ruthruff, J.R. and S. Elbaum and G. Rothermel, 2006. Experimental program analysis: A new program analysis paradigm. Proceedings of the International Symposium on Software Testing and Analysis, July 17-20, Portland, ME, USA., pp: 49-60.

Stewardson, D.J., C. Hicks, P. Pongcharoen, S.Y. Coleman and P.M. Braiden, 2002. Overcoming complexity via statistical thinking: Optimising genetic algorithms for use in complex scheduling problems via designed experiments. Proceedings of Tackling Industrial Complexity: The Ideas that Make a Difference, April 9-10, Downing College, Cambridge, London, pp: 275-290.

Totaro, M.W. and D.D. Perkins, 2005. Using statistical design of experiments for analyzing mobile ad hoc networks. Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Oct. 10-13, Montreal, Canada, pp: 159-168.

Whiting, D.G., Q. Snell, R.R. Nichols, M.L. Porter and K. Tew *et al*., 2004. Complex performance analysis through statistical experimental design: An evaluation of parameters associated with speed in parallel phylogenomics. Hawaii International Conference on Computer Science, January 2004, pp: 615-629.

Zheng, G., G. Gupta, E. Bohm, I. Dooley and L.V. Kale, 2010. Simulating large scale parallel applications using statistical models for sequential execution blocks. Proceedings of the 16th International Conference on Parallel and Distributed Systems (PDS'10). http://charm.cs.uiuc.edu/papers/10-15.shtml.