Causal Models for Parallel Performance Analysis

Jan Lemeire[1], Sam Maes[2], Erik Dirkx[1]

[1]Parallel Systems lab,
http://parallel.vub.ac.be
[2]Computational Modeling lab,
http://como.vub.ac.be/
Vrije Universiteit Brussel,
Pleinlaan 2, 1000 Brussels, Belgium
{jlemeire, erik}@info.vub.ac.be
sammaes@vub.ac.be

Abstract

This paper proposes causal models for enhancing the performance analysis of parallel programs. Causal models make the relations among the variables of interest explicit. By applying this statistical methodology to performance analysis we envisage to further automate the performance modeling task and to present the user a clear and understandable analysis. It is a flexible approach, since new environment variables can easily be integrated and performance can be estimated with incomplete knowledge. Independency among variables is the key information, therefore it can help the construction of a performance model that separates application and system dependency.

Keywords: Performance analysis, parallel performance, causal models, causality, performance tools.

1. Introduction

For efficient parallel processing, the developer must master various aspects influencing the performance, ranging from high-level software issues to low-level hardware characteristics. The performance analysis is nowadays supported by various performance tools that automatically instrument code, collect performance data during program execution and provide a post-mortem analysis that relates the hardware performance data to the program code areas (SCALEA, VAMPIR, KappaPi, Pablo, AIMS, etc.). Current challenges are further automation, tackle complex situations (eg. in GRID environments) and to give the software developer understandable results with a minimum of learning overhead [APART working group: http://www.fz-juelich.de/apart].

We believe that causal models are a useful representation for the performance analysis, capture the relevant questions about performance and can guide statistical analysis of experimentally retrieved performance data

Causality is a part of the domain of statistics that studies causal relations. It is widely used in social sciences like economy and sociology, biology, machine learning,… However, causality is a highly debated topic among statisticians [Pearl 2000], since a causal relation represents more than the basic statistical correlation among probabilistic variables. However, causality cannot be observed directly, only statistical correlations can, where “a correlation being the observational shadow of the underlying causal process” [Shipley 2000].

A causal model consists of a graph, which offers a clear and understandable representation of the model and shows the direct relations among variables. These relations are closely related to the underlying process. Therefore, causal models are closer to the human mental models than other more traditional mathematical or statistical models [Pearl 2000]. The models can be used for prediction, but also for reasoning about the effects of changes - called interventions - of the model [Pearl 2000].

Section 2 defines causal models, in section 3 we explore how causal models about performance can be constructed. Section 4 is dedicated to the automatic discovery of causality in experimental data. Section 5 explains additional properties of the causal approach and section 6 gives an overview of its utility.

2. Causal models

Figure 1 shows an example of a medical causal model, in which diseases (middle row) are influenced by environmental factors (top row) and generate symptoms (bottom row). All relations are probabilistic and mean that chances of a variable increase if one of its ascendants increase.

A causal model consists of a Directed Acyclic Graph (DAG) over a set $V=\{V_1,\ldots, V_n\}$ of vertices, representing variables of interest, and a set E of directed edges, or arrows, that connect these vertices. The interpretation of such a graph has two components, probabilistic and causal [Tian 2002]. The probabilistic interpretation view the arrows as representing probabilistic dependencies among the corresponding variables, and the missing arrows as representing conditional independence assertions:

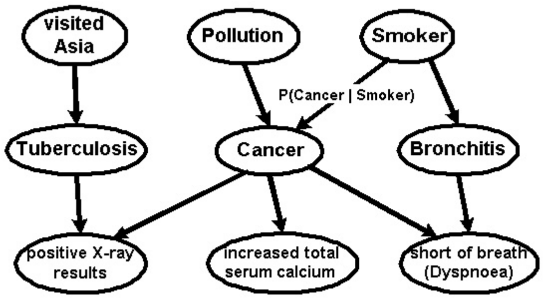


Figure 1. Medical Example of a Causal Model

Each variable is independent of all its non-descendants given its direct parents in the graph. In the example of Figure 1, having positive X-ray results depend on being a smoker, but not anymore if it is known if the patient has cancer, than both variables become independent.

The assumptions amount to asserting that the joint probability function $P(v)=P(v_1,\ldots, v_n)$ factorizes according to the product

$$P(v)=\prod_i P(v_i \mid pa_i) \qquad (1)$$

Where pa_i denotes the set of parents of variable v_i in the graph.

The causal interpretation views the arrows as representing causal influences between the corresponding variables. In this interpretation, the factorization of (1) still holds, but the factors are further assumed to represent autonomous data-generation processes, that is, each conditional probability $P(v_i|pa_i)$ represents a stochastic process by which the values of V_i are chosen in response to the values pa_i, and the stochastic variation of this assignment is assumed independent of the variations in all other assignments. Moreover, each assignment process remains invariant to possible changes in assignment processes that govern other variables in the system. This modularity assumption enables us to predict the effect of interventions, whenever interventions are described as specific modifications of some factors in the product of (1).

Probabilistic independence of X and Y upon conditioning on Z, denoted $Ind(X, \mathbf{Z}, Y)$, implies that

$$Ind(X, \mathbf{Z}, Y) \Leftrightarrow P(X,Y|Z)=P(X|Z).P(Y|Z). \qquad (2)$$

For linear dependencies among the variables, this dependency can be measured by its *correlation coefficient*. The arrows represent *direct* causal relations, meaning that they cannot become probabilistically independent upon conditioning on some other set of vertices.

A causal model implies all relations to be of *causal* nature. A causal relation is an irreflexive, transitive and asymmetrical (rain creates mud, but mud will not create rain) relation. It also has the properties of *productivity* (the effect is 'produced' by the cause) [Bunge 1979, p. 48], *locality*, it obeys the markov condition (for model $A \rightarrow B \rightarrow C$, if B is blocked, than A doesn't cause C) and represents a stable and *autonomous* physical mechanism ("which is conceivable to change one relationship without changing the others") [Pearl 2000]. These properties make it possible to reason about *interventions* (Pearl therefore introduced the *do(x)* operator) and in some cases answer questions like "what if I increase the cache memory" or "what if I use another sort method", without actually performing these actions.

2.1. Information-Theoretic View

Probabilistic dependency can be rewritten, in terms of Information Theory, as the mutual information $I(X,Y)$ of X and Y, which is the reduction in uncertainty or entropy of X when knowing Y [Cover 1991]:

$$I(X:Y)=H(X)-H(X\mid Y) \qquad (3)$$

It is zero, when both variables are independent. The entropy of a discrete random variable X with alphabet A and probability mass function $p(x)$ is defined as

$$H(X)=-\sum_{x\in A} p(x)\log p(x) \qquad (4)$$

Entropy is defined as the amount of uncertainty of a stochastic variable. The conditional entropy $H(Y|X)$ is defined as

$$H(X\mid Y)=-\sum_{y\in B} p(y)H(X\mid Y=y) \qquad (5)$$

where B is the alphabet of random variable Y. The mutual information can then be rewritten as

$$I(X:Y)=\sum_{x\in A}\sum_{y\in B} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \qquad (6)$$

Conditional independence, $I(X;Y|Z)$, is defined in the same way.

3. Causal Models of Performance

Our research investigates how a performance analysis can benefit from the approaches offered by statistics, and more specific causality theory. We believe that traditional statistics can be beneficial in the in-depth investigation of experimental data; we come back to this point later. Many relations in performance models are of causal nature.
If we want to detect program performance problems, then we seek the 'true' reasons of bad performance. See for example performance analysis tools as Kojak [Mohr 2003], Paradyn or KappiPi [Espinosa 1998], which searches for inefficiency patterns, like 'blocked sender', 'lack of parallelism', 'barrier problems',… These are questions about causality.

However, if we define the efficiency of a parallel program as $E = S/p$ (with S the speedup and p the number of processors), then we cannot speak about a causal relation. E is not 'caused' by S or p. This is a functional relation, S could also be derived from E, namely $S=pE$. But this does not hold for causal relations (note: naturally S and E are correlated, but this is because they share the same causes) When $A \rightarrow B$ holds, we can write $B = xA$ or $A = B/x$, but the information that B is generated by A is lost. The $\rightarrow$ operator expresses more than a pure functional relation, which we will indicate in the diagrams with just a straight line.

Figure 2 represents a causal model of performance related data concerning a quicksort running on a sequential computer. It represents a first-order approximated performance model for the sequential runtime T_{comp} of a quicksort. *#op* is the number of basic compare-swap operations, which is determined by the array size *n* and the initial *order* of the elements. The compare and swap statements correspond to a number of basic instructions $\#instr_{op}$, that together with the processor's clock frequency f_{clock} determine the time for 1 operation T_{1op}. In the example of Fig. 2, once we know the number of iterations *#op*, T_{comp} becomes independent of *n* or the *initial order*. This is a *reduction* of the dependency complexity of the model.

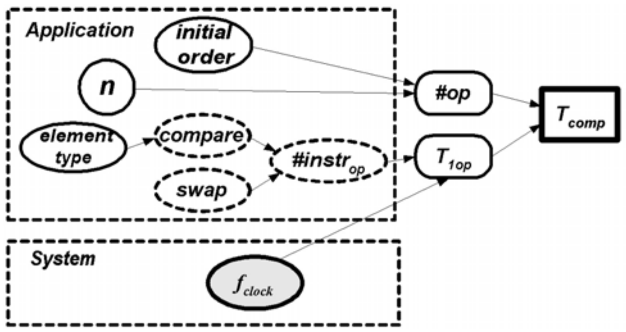


Figure 2. Simplified Causal Performance Model of Quicksort.

3.1. Parallel Performance Analysis

The models in the analysis of parallel applications can also be written as a causal diagram, as shown in Figure 3 [Lemeire 2004]. Three main phases of the parallel runtime T_{par} are identified: computation, communication and idling.

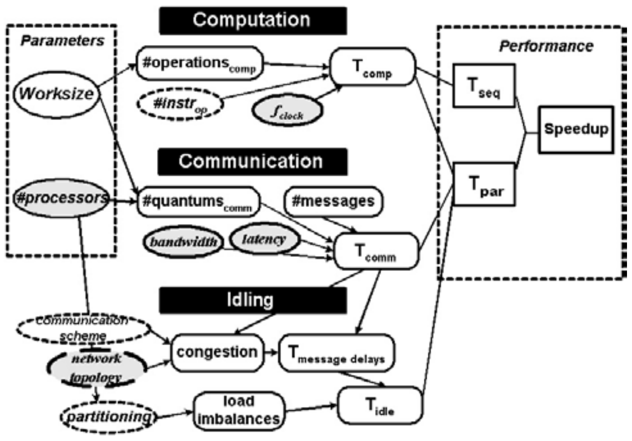


Figure 3. A Causal Model of Parallel Performance.

The parallel performance metrics uses an overhead quantification based on the *lost-cycle approach* [Crovella '94]. It aims at attributing each part of the overhead runtime (the so-called lost cycles) to the purpose it was spent. This can be viewed as questions about causality. Program parameters and system characteristics (grey ovals) influence the quantities of the overheads and therefore are also of causal nature.

3.2. Models for Network Performance

In the study of network performance - for analysis or prediction - communication delays should be attributed to the different steps of the communication process, like machine latency, transfer time, network contention, flight time, etc [Badia 2003]. Correct understanding of the correct origins – or causes – is indispensable. However, when low level issues, like specific protocol behavior, window delays or chatter are the reasons, the task of identifying them becomes extremely difficult [NetPredict 2003], since they are not always fully understood and cannot be measured directly. The understanding of these message delays can also be viewed as the search for causes.

4 Discovery of Causal Relations

The construction of causal models out of experimental data is widely investigated since the pioneer work of both Verma and Pearl [1991] and Spirtes, Glymour and Scheines [1993] in the early 90's. Algorithms are based on the *d-separation* criterion, which gives the necessary and sufficient conditions for two vertices in a graph to be probabilistically independent upon conditioning on some other vertices. This indicates that there is no direct causal relationship and no edge in the graph, as explained in section 2 [Pearl 2000, pp. 16-19]. Various tools that implement these algorithms exist, like TETRAD [http://www.phil.cmu.edu/projects/tetrad/] or PNL [http://www.intel.com/research/mrl/pnl/]. Most algorithms work for discrete variables. For continuous variables however, the correct model is expected to be multivariate normal and approximately linear, since they use statistical correlations for testing conditional independence. Performance models are not always linear and a mixture of discrete and continuous variables
We applied the construction algorithms of TETRAD on experimental data gathered from a quicksort of an array of n elements of different types (short, integer, float and double). The performance results for the computation time T_{comp} are shown in Figure 4.

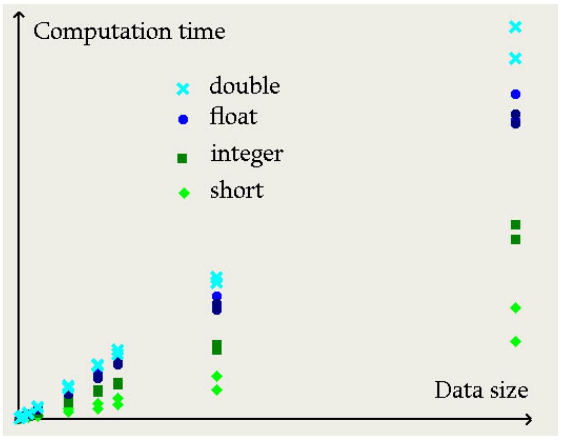
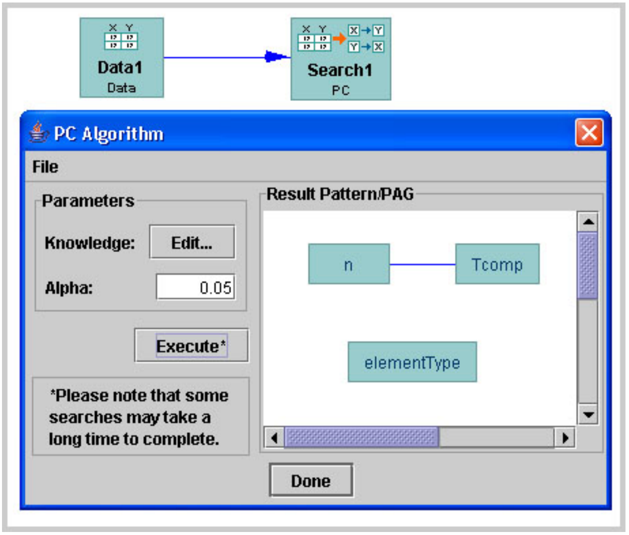


Figure 4. Quicksort runtime versus array size for different element types.

The impact of both parameters is obvious. However, the causal discoveries algorithm of TETRAD only find a relation between n and T_{comp}, as can be seen for the PC algorithm in Figure 5. This can be explained by the diagrams of Figure 6, that show that the array size dependency is quasi linear, while the runtime dependency of the element type is less clear.



*Please note that some searches may take a long time to complete.

Figure 5. Causal structure learning out of experimental data with the PC algorithm of Tetrad 4.3.

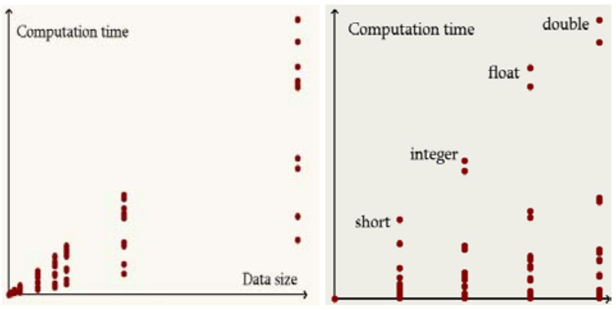


Figure 6. Quicksort runtime versus array size (left) and element type (right).

We will have to enlarge the scope of the algorithms, by using a more general test of conditional independence, as given by information-theoretic definitions (see 2.1).

5 Advanced Properties of Causal Performance Models

For successful application of causal models in the study of parallel performance, two more requirements should be met, namely flexibility and the possibility to reduce submodel dependencies.

5.1. Flexibility

At each stage, it should be possible to refine models with extra information [Lemeire 2004]. Figure 7 shows an extended version of the quicksort performance model of Figure 2. Memory overheads, denoted by T_{memory}, were added to the model, they are caused by the application's *data size*, *memory usage* and the processor's *memory capacity* and *bandwidth*.

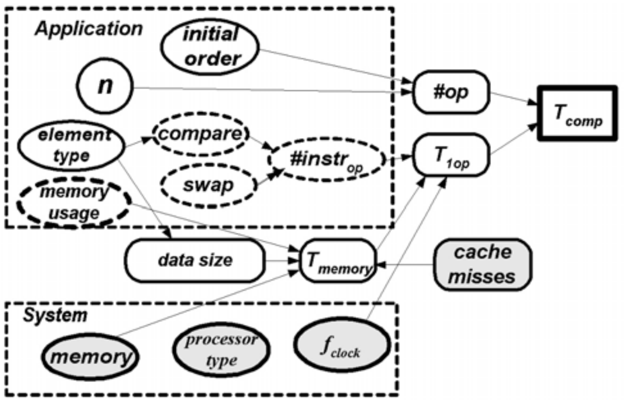


Figure 7. Detailed Causal Performance Model of Quicksort.

The modeler can integrate additional information into the model, like the *cache misses* (measured with for example the PAPI tool for accessing hardware counters on microprocessors [Browne 2001]) or the *processor type* , as shown in Figure 7. Through statistical analysis, the dependencies with the other variables can be found and the predictive qualities of this extra information can refine the performance model.
On the other hand, not all variables should be known for performance prediction. By the use of statistics, the expectancies can be calculated for unknown variables.

In this way, it should be possible to create flexible, hierarchical models.

5.2. Separation of application and system dependency

The ultimate goal of the performance analysis is to be able to predict the runtime of an application on any system without having to run tests on it. This requires however that there exist independent application and system characteristics and a functional relation to calculate the resulting performance. This is the case in the simplified example of Fig. 2, where these characteristics are *#op*, $\#cycles_{op}$ for the application and f_{cl} for the system. The equation becomes simply:

$$T_{comp}=\#op.\#cycles_{op}.1/f_{cl}. \qquad (7)$$

This first-order approximation however only holds for small problem sizes. When memory overheads come into play, as in the extended model of Figure 7, the separation is much less trivial [Snavely 2002].
This requires the construction of submodels that are independent of either algorithm or system. This dependency information is exactly what causal models represent.

6 Utility

The utility of causal models is twofold: they should support the modeler as well as the user.

6.1. Support of the performance modeling process

1. *Model validation*: validation of the (in)dependency assumptions made by the modeler.
2. *Reuse of autonomous relations:* for example, the statistical analysis of several experiments on a certain network would give an overall model for the communication time versus the data size.
3. *Detection of abnormal, unexpected dependencies*, like non-homogeneous situations or high overheads. The statistical model of point 2 can be used to warn for communication delays above the average, in for example GRID environments [Balis 2004].
4. *Flexibility* is a necessary requirement, as discussed in the previous section.

6.2. Presentation of a clear performance report

In order to understand complex situations with many variables and dependencies, a structured representation of the relations is required. Causal models furthermore correspond to physical mechanisms and enable the filtering of the relevant information: the statistical analysis will reveal the impact of every factor, so that the most influential factors can be highlighted.
Besides the explanational facilities, the causal models could be exploited to reason about the performance and answer questions like: Which part of the application gives space for adequate optimization? What is the most efficient upgrade of the system?

7. Conclusions

Recent developments by statisticians and computer scientists in the field of causality show promising results for being applied in the construction of performance models. Causal modeling and the corresponding statistical analysis make explicit what is done by the scientist when analyzing performance. This makes further automation possible.

However, current tools do not support the analysis of models as we encounter in the field of (parallel) performance analysis. Our current ongoing work therefore focuses on the extension of current algorithms for discovery of causal relations.

8. References

Badia, Rosa M. et all., DIMEMAS: Predicting MPI applications behavior in Grid environments, Workshop on Grid Applications and Programming Tools (GGF8), 2003.

Balis, B. et all. Performance Evaluation and Monitoring of Interactive Grid Applications, In Proc. of the 11th EuroPVM/MPI Conference, Budapest, Hungary, Sept. 19-22, 2004.

Browne, S., Dongarra, J.J., Garner, N., Ho G., and Mucci, P. *A Portable Programming Interface for Performance Evaluation on Modern Processors*, International Journal of High Performance Computing Applications, 14:3 (Fall 2000), pp. 189-204.

Bunge, Mario. *Causality and Modern Science*, third revised edition, Dover Publications, New York, 1979.

Cover, Thomas M. and Thomas, Joy A. *Elements of Information Theory*, Wiley, 1991.

Crovella, M. E. and Leblanc, T.J.: Parallel Performance Prediction using Lost Cycles Analysis. In: Proc. of Supercomputing ’94, IEEE Computer Society (1994).

Espinosa, A., Margalef, T. and Luque, E., Automatic Detection of PVM Program Performance Problems. In Proc. of the 5th PVM/MPI Conference, pp. 19-26, 1998.

Lemeire, J., Crijns, A., Crijns, J. and Dirkx, E., A Refinement Strategy for a User-Oriented Performance Analysis. In Proc. of the 11th EuroPVM/MPI Conference, Budapest, Hungary, Sept. 19-22, 2004.

Mohr, B., and Wolf, F. KOJAK - A Tool Set for Automatic Performance Analysis of Parallel Programs. *Euro-Par Conf.* 2003: 1301-1304.

NetPredict, Inc. Common Mistakes in Performance Analysis, White Paper, NetPredict Inc, March 2003.

Pearl, J. *Causality. Models, Reasoning and Inference*. Cambridge University Press, Cambridge, 2000.

Shipley, Bill. *Cause and Correlation in Biology*, Cambridge University Press, 2000.

Snavely, A. et all., *A framework for performance modeling and prediction*. In Proc. of the 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland pp. 1-17, 2002.

Spirtes, P., Glymour, C. and Scheines, R. Causation, prediction and search. New York, Springer-Verlag, 1993.

Tian, J. and Pearl, J. A general identification condition for causal effects. In Proc. of the national conference on Artificial intelligence, Edmonton, Canada, pp. 567-573, 2002.

Verma, T. and Pearl, J. Equivalence and synthesis of causal models. In Proc. of the 6th workshop on uncertainty in Artifical Intelligence, Cambridge, MA, 1991.