Causal Models for Parallel Performance Analysis

Jan Lemeire, Erik Dirkx

Parallel Systems lab, Vrije Universiteit Brussel,
Pleinlaan 2, 1000 Brussels, Belgium
{jlemeire, erik}@info.vub.ac.be
http://parallel.vub.ac.be

4th PA3CT Symposium, September 2004, Edegem, Belgium

Abstract

This paper proposes causal models to enhance the performance analysis of parallel processing. Causal models explicitly denote the relations among the variables involved. This makes it possible to automate the modeling task as well as to present the user a clear and understandable performance analysis. It is a flexible approach, since new environment variables can easily be integrated and performance can be estimated from incomplete knowledge. Since independency among variables is the key information, it can help the construction of a performance model that separates application and system dependency.

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 end-user tools as PAPI for accessing hardware counters on microprocessors [Browne 2001], as well 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.). The current challenge is 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 relations are closer to the human mental model than other more traditional mathematical or statistical models [Pearl 2000]. Moreover, we believe there is a challenge towards more in-depth statistical analysis of experimentally retrieved performance data. However, causality is a highly debated topic among statisticians [Pearl 2000], since it represents more than a statistical correlation. "A correlation being the observational shadow of the underlying causal process" [Shipley 2000].

2. Causal models

Causal models are expressed with Directed Acyclic Graphs (DAG), see Figure 1, which 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 processors clock frequency f_{clock} determine the time for 1 operation T_{1op}.

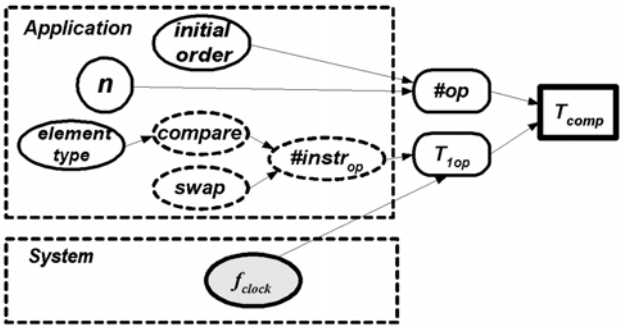

Figure 1. Simplified Causal Performance Model of Quicksort.

A causal model represents first of all a *Bayesian network*, where the joint probability distribution $P(x_1, \ldots, x_n)$ in which all variables are related, is reduced when each variable depends on just a small subset of other variables. The probability of a variable can then be calculated by the probabilities of its parents in the model.

The *d-separation* criterion gives the conditions for two vertices to be *probabilistic independent* upon conditioning on some other vertices [Pearl 2000, pp. 16-19]. In the example of Fig. 1, 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.

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

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

Or, in terms of Information Theory, the *mutual information* $I(X,Y)$ of X and Y is 0 when there is no reduction in uncertainty (entropy) of X when knowing Y [Cover 1991]:

$$I(X;Y) = H(X) - H(X|Y)$$

For lineair dependencies among the variables, this dependency can be measured by its *correlation coefficient*.

A causal model however represents more than a Bayesian model. It 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 answer questions like "what if I increase the cache memory" or "what if I use another sort method".

3. Causal Performance Models

Fig. 1 represents a causal model of performance related data concerning a quicksort running on a sequential computer. The models in the analysis of parallel applications are similar, see [Lemeire 2004]. We defined the parallel performance metrics using an overhead quantification based on the *lost-cycle approach* [Crovella '94].

3.1. Flexibility

The model can always be *refined* [Lemeire 2004], for example, by adding memory overheads to the quicksort model, see Fig. 2, where T_{memory} is caused by the application's *data size*, *memory usage* and the processor's *memory capacity* and *bandwidth*.

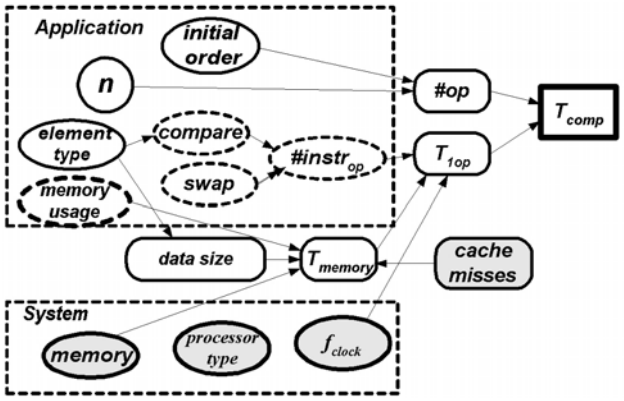

Figure 2. Extended Causal Performance Model of Quicksort.

Furthermore, the modeler can integrate additional information into the model, like the measured *cache misses* or the *processor type* (Fig. 2). Through statistical analysis, the dependencies with the other variables are found (Eg. Between *element type* and *processor type*) and the predictive qualities of this extra information can enhance the performance model.

On the other hand, not all variables should be known for performance prediction. The statistical expected value can be used for unknown variables.

3.2. Separate application and system performance dependency

The ultimate goal of the performance analysis is to be able to predict the runtime of an application on an unknown system. This requires however that there exist independent application and system characteristics and a simple functional relation to calculate the resulting performance. This is the case in the simplified example of Fig. 1, where these characteristics are $\#op$, $\#cycles_{op}$ for the application and f_{cl} for the system. The equation is:

$$T_{comp}=\#op.\#cycles_{op}.1/f_{cl}.$$

This first-order approximation however only holds for small problem sizes. When memory overheads come into play, as in the extended model of Fig. 2, the separation is much less trivial [Snavely 2002].

4 Utility

The construction of causal models out of experimental data is widely investigated [Pearl 2000, Shipley 2000], and various tools exist, like Tetrad [http://www.phil.cmu.edu/projects/tetrad/] or PNL [http://www.intel.com/research/mrl/pnl/]. We are

constructing a parallel performance tool (EPPA: http://parallel.vub.ac.be/eppa) that gathers experimental data and automatically analyzes the data as discussed in the previous section. Such a tool envisages the support of the modeler and the user.

A. For supporting the performance modeling process

1. *Model validation*: validation of the (in)dependency assumptions made by the modeler.
2. *Reuse of autonomous relations* (eg. 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* (eg. non-homogeneous situations, high overheads…)
4. *Flexibility*: as discussed in the previous section, no information is lost.

B. For presenting the user a clear performance report

1. *Structuring the variables*: causal relations correspond to physical mechanisms.
2. *Filtering relevant information*: the model will reveal the impact of every factor, thus the most influential factors can be highlighted.
3. *Reasoning about interventions*: Eg. Which part of the application gives space for adequate optimization? What is the most efficient upgrade of the system?

5. Conclusion

Causal modeling and the corresponding statistical analysis make *explicit* what is done by the scientist when analyzing the performance of the software – hardware match. This makes further automation possible. However, the limitations to the automation are yet still unclear.

6. References

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).

Lemeire, J. *A Refinement strategy*… EuroPvm, 2004.

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.