Evaluating a Data Removal Strategy for Grid Environments Using Colored Petri Nets

Nikola Trčka, Wil van der Aalst, Carmen Bratosin, and Natalia Sidorova

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{n.trcka,w.m.p.v.d.aalst,c.c.bratosin,n.sidorova}@tue.nl

Abstract. We use Colored Petri Nets (CPNs) for the modeling and performance analysis of grid architectures. We define a strategy for the optimization of grid storage usage, based on the addition of data removal tasks to grid workflows. We evaluate the strategy by simulating our CPN model of the grid. Experiments show that the strategy significantly reduces the amount of storage space needed to execute a grid application.

1 Introduction

Grid computing has emerged as a powerful platform for executing data and computation intensive applications. Several grid architectures have been proposed to orchestrate available resources (e.g. [1,5,2]). However, as the complexity of grid applications continuously increases, there is always a need for new solutions. These solutions are typically first evaluated on a simulation model. In this paper we propose Colored Petri Nets (CPNs) [6] for the modeling and simulation of grid architectures.

Grid simulation has been an active area of research for quite some time, and several grid simulators have been developed [7,4]. There are, however, several advantages in using CPNs for this purpose: 1) CPNs are graphical, hierarchical, modular, and have a formal semantics[1]. They are executable and thus can model the dynamics of the grid as well. 2) CPNs are supported by CPN tools [6], a powerful framework for modeling, verification and performance analysis. 3) Petri nets have been extensively used for many years to model and analyze concurrent systems. Simple reuse of ideas makes grid modeling a relatively easy task. 4) Most grid workflow languages can be easily converted to Petri nets.

The grid architecture we model is reasonably generic, suitable for executing grid applications that are computationally intensive and manipulate (large) data. It can be seen as a computational grid in which data also plays an important role. We believe that our model covers all relevant aspects with enough detail, and that it can be used to discover trends that remain valid in more complex settings.

Every task in a grid workflow typically specifies the set of its input data files and the data that it generates. We frequently see cases where some data is

[1] The last feature is very relevant and directly used in this work.

T.P. Baker, A. Bui, and S. Tixeuil (Eds.): OPODIS 2008, LNCS 5401, pp. 538–541, 2008.

only used at one region of a workflow. This data, although not needed after a certain point in time, stays on the grid until the complete workflow is finished. In an environment with reliable resources this is a waste of storage space and an optimization strategy is needed. In this paper we propose one such strategy, in form of a method that inserts data clean-up tasks to a workflow, at the points from which no further tasks access this data. We perform simulations on our CPN model of the grid, and show that the amount of storage space an application uses during execution is indeed significantly reduced when our strategy is applied.[2]

2 Modeling Grid Architectures

Our grid architecture consists of three layers. On the top is the *application layer*, a workflow environment in which the users describe their applications. On the bottom is the *fabric layer* consisting of grid nodes connected in a network. In between is the *middleware layer*, responsible for the allocation of resources and data handling.

Fig. 1 shows the CPN module of the scheduling system. The module illustrates the complexity of, and the level of detail covered by, our CPN model. It also shows the descriptive power of CPNs, as only basic knowledge of Petri nets is needed to understand how the scheduler works.

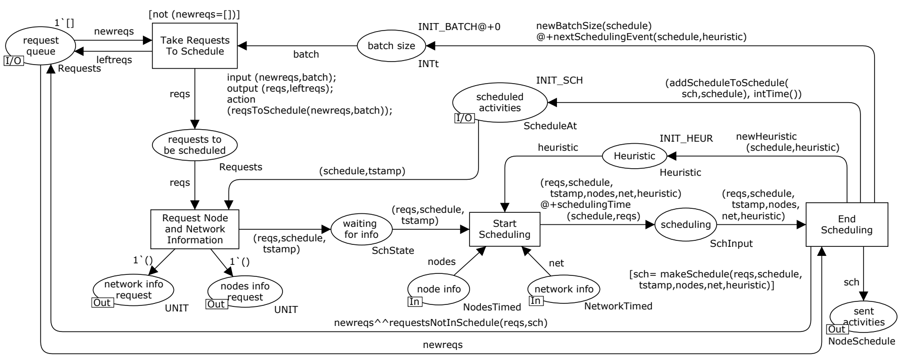

Fig. 1. CPN model of the scheduler

3 Data Removal Strategy

Our optimization strategy is based on the insertion of data clean-up tasks at the points from which this data is no longer needed. Since data elements can be (re)used in loops, in parallel or in alternative branches, the main challenge is to identify these points. However, Petri nets provide ways to achieve this easily. We present the method on the basis of an example.

[2] This short paper presents the main ideas only; full details are given in [8].

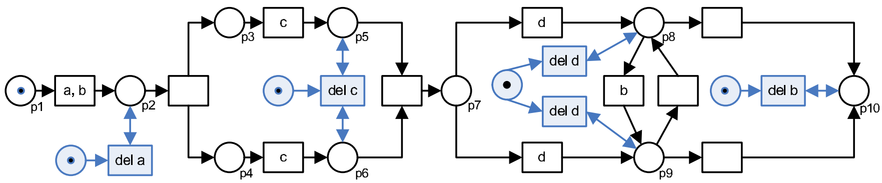

Fig. 2. Data clean-up tasks added to a workflow

Table 1. Percentage of storage space freed

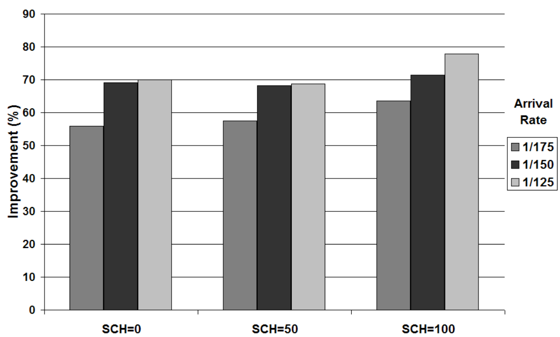

Fig. 2 shows a (sound[3]) grid workflow (ignore the gray elements for the moment), where a, b, c, and d are data elements shared among the tasks. Our method introduces a clean-up task for each element, with one guard place to ensure that the removal request is issued only once. These additions are illustrated in gray color in Fig. 2. The element c is thus deleted when there is a token in place p_5 *and* a token in place p_6. The element d is deleted when there is a token in p_8 *or* a token in p_9. While a can be deleted immediately after the first task, b can only be deleted at the end, as it can be used in the loop between p_8 and p_9.

We evaluate the strategy by means of simulation, using our CPN model. The testbed consists of nine grid nodes, fully connected in a homogeneous network. The input workflow is fixed, but investigated for the case-arrival rates ranging from 1/175 to 1/125. We use periodic Min-Min scheduling with the period varied from 0 to 100.

The results in Table 1 show the percentage of storage space that becomes available when the strategy is applied. The improvement is greater for longer scheduling periods and for higher arrival rates. This is expected, as longer periods enable more effective prioritization of removal tasks, and more cases result in longer unnecessary space occupation.

[3] Soundness property [3] is a sanity check for workflows. It ensures that the workflow can always complete, and that it has no dead transitions.

4 Conclusion

We modeled a grid architecture in terms of Colored Petri Nets. The model is formal, graphical and executable, offering an unambiguous view of how different parts of the grid are structured and how they interact. The model is suitable for performance analysis, it is fully adaptable and extendible.

To solve the problem of data occupying the grid storage space unnecessarily long, we introduced a method for the addition of data clean-up tasks to grid workflows. We evaluated this method by conducting a simulation experiment using our CPN model of the grid. The results showed that the required storage space could be reduced by as far as 80%.

References

1. Condor Project, `http://www.cs.wisc.edu/condor/`
2. K-WfGrid web site, `http://www.kwfgrid.eu`
3. van der Aalst, W.M.P.: Verification of Workflow Nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
4. Buyya, R., Murshed, M.M.: GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. Concurrency and Computation: Practice and Experienc 14(13-15), 1175–1220 (2002)
5. Deelman, E., Singh, G., Su, M., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C., Katz, D.S.: Pegasus: A framework for mapping complex scientific workflows onto distributed systems. Sci. Program. 13(3), 219–237 (2005)
6. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer 9(3-4), 213–254 (2007)
7. Legrand, A., Marchal, L., Casanova, H.: Scheduling Distributed Applications: the SimGrid Simulation Framework. In: 3rd International Symposium on Cluster Computing and the Grid (CCGRID 2003), Washington, pp. 138–146. IEEE, Los Alamitos (2003)
8. Trčka, N., van der Aalst, W.M.P., Bratosin, C., Sidorova, N.: Evaluating a Data Removal Strategy for Grid Environments Using Colored Petri Nets. Technical report, Eindhoven University of Technology (2008), `http://w3.win.tue.nl/nl/onderzoek/onderzoek_informatica/computer_science_reports/`