Dynamic Control in Workflows of Scientific Computations with Monitoring of Activities Global States

Damian Kopański[1], Marek Tudruj[1,2], Janusz Borkowski[1]
[1] Polish-Japanese Institute of Information Technology,
86 Koszykowa Str., 02-008 Warsaw, Poland
[2] Institute of Computer Science, Polish Academy of Sciences,
21 Ordona Str., 01-237 Warsaw, Poland
{damian, tudruj, janb}@pjwstk.edu.pl

Extended Abstract

Workflow applications consist of parallel programs executed in cluster environments. The execution of these programs must be coordinated. In our design, a special global control infrastructure is used for these purposes. Each cluster-level parallel program has an associated control process called a synchronizer. Constituent processes report their local states to the local synchronizer. The synchronizers compute program level global states and evaluate control predicates on them. A programmer defines the predicates. Based on the evaluation result, control signals can be sent to processes to communicate control decisions. As a result of control signal reception, actions defined by a programmer can be activated influencing program behavior in an asynchronous manner. This control method de-couples the control aspects solved using a high-level abstraction, from the computational code. The PS-GRADE graphical program design environment, developed as an enhancement of the original P-GRADE system, supports the described control method. Parallel applications, implemented using the PS-GRADE system and run in clusters, can be executed in a coordinated fashion to form a workflow-level multi-application [3]. To achieve this, workflow-level synchronizers are introduced. They gather state reports from application synchronizers construct workflow-level global states, evaluate predicates on them and send control signals to applications.

We propose a generalized workflow design system, which extend basic workflow control paradigms. The system is constructed with the use of the synchronizers, which allow for dynamic behavior of the resulting workflow. Many researchers have investigated dynamic features in workflows e.g. [3,4]. In our system, the notion of dynamic workflows denotes modifications of control flow between constituent activities and modifications of internal behavior (normally completely opaque) of the activities.

Our approach to workflow applications control relies on monitoring states on constituent parallel applications. Application states are constructed from process reports about their local states. A set of user-defined predicates is evaluated on the global/regional states. When a predicate is satisfied, then control signals are dispatched to chosen processes. To prevent passive waiting for control decisions, the signals are asynchronously handled – they can activate a signal-associated program code or they can cancel current computation causing a process to ignore a section of

the program code immediately upon reception. Signals can carry some data, which can be used by the activated code and which can be introduced into the ongoing main computations.

The application states we talk about are Strongly Consistent Global States (SCGS), which are parallel process states that occurred at the same time. This approach requires, that timestamps determined with a known accuracy are attached to process state reports. In many cases, the use of timestamps (and the necessary local clock synchronization) can be omitted and Observed Global States can be used instead of SCGS. This situation is frequent in workflow control.

The PS-GRADE graphical parallel program design system incorporates the described control mechanisms [2]. The synchronizer process has special communication ports to receive state reports and to send control signals. It works according to the scheme shown in Fig. 1. It waits for a state report from a process and checks if a new consistent/observed global/regional state has been reached. If so, it further checks, which predicates should be evaluated on the encountered state and evaluates them. Separate predicates can be defined for global and regional states for each defined region (a region is a subset of the full set of application processes). A predicate code is expressed in a form of a control flow diagram of a similar type as a process code.

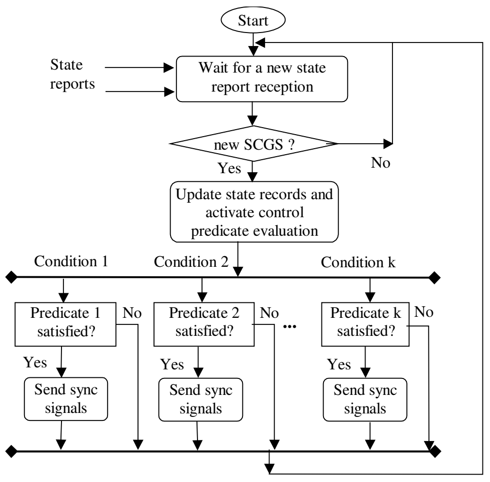

Fig. 1. Internal control of a synchronizer

Using PS-GRADE to implement component parallel applications, we build Grid-level multi-application with the use of special extensions of the PS-GRADE system. The component applications are treated as workflow activities. Additional layer of synchronizers was introduced. Standard application-level synchronizers report application state to Grid-level synchronizers. The Grid-level synchronizers compute control predicates on these states and send signals to influence the inter-application control flow, e.g. by blocking or activating a path in a dataflow scheme. The main advantage of our approach over a usual workflow system is that the workflow activities show modifiable behavior, which can be managed externally

There are a number of areas where the proposed workflow control mechanisms are useful and cannot be replaced by a standard workflow in an efficient and convenient way, especially for more complicated control flow patterns. We can subdivide known workflow instances into static workflows and dynamic workflows. Static workflows contain activities (tasks), which are atomic. In a static workflow, the functional contents do not change during workflow execution. Once started, static workflow activities are executed until completion or their execution can be canceled. All control flow decisions are taken at the beginning or after completion of the activities. In static workflows, final states of activities (tasks) between their activations are examined to influence the control flow in the workflow.

Dynamic workflows should allow for modifications of control flow and functional contents of activities during workflow execution, based on global state analysis of subsets (or the entire set) of activities (tasks) belonging to the workflow. The workflow control method that we propose is based on global state analysis of subsets (or the entire set) of activities (tasks) belonging to the workflow. It is done with the use of the control infrastructure provided by synchronizers. In our approach to dynamic workflows, the notion of dynamic workflows covers modifications of control flow between constituent workflow activities and modifications of internal functioning of activities during workflow execution. The modifications are asynchronous in respect to activities activation/termination. Dynamic insertions of new activities (workflow blocks) are not provided. Instead of that, new actions inside workflow activities can be asynchronously launched as a result of programmed control based on global workflow state analysis.

Dynamic workflow can accelerate a distributed search expressed as a workflow executed in the Grid. In a parallel multi-application performing a distributed search, each application instance (an element of a workflow) gets a fraction of the solution space to process. In a classical approach every instance completes independently and passes its results. Our dynamic workflow mechanisms allow for a communication between the running elements. A found solution is immediately reported by a local synchronizer to a workflow-level synchronizer. Then, the new solution is distributed to all workflow elements using asynchronous notification. Thanks to that, the search performed locally in each application instance can continue while taking into account solutions found by other instances, thus omitting search space regions which cannot contain solutions better then already known (ex. branch-and-bound search).

One of important problems in scientific computations with the control expressed as workflow is dynamic optimization of the use of computing resources in the system during workflow execution. The most important kind of such optimization seems to be load balancing of processors in clusters belonging to Grids, which leads to the reduction of execution time of workflow activities and as a consequence in a heuristic manner reduces the total execution time of their programs. The infrastructure of the proposed control method – state reporting, global state analysis, control signal dispatching, asynchronous reaction on signals – suites very well as a framework for load balancing implementations. A user needs to define a load measure to be sent as the current process state, a predicate to analyze the load imbalance, and the code activated by control signals. Each of the mentioned components can be easily changed if necessary. So, the proposed control method can be efficiently used as a base for implementing a user defined load balancing strategy in a parallel application.

We can identify two methods of organizing load balancing in workflows. The first method which can be called “migrative” is based on the analysis of global loads states in clusters which execute activities of the scientific computation workflow by Grid-level synchronizers. They work out the global predicates on load states and stimulate activity migration from too much loaded parts of the Grid to some other less loaded parts or stimulate spawning of new activities in such less loaded parts of the Grids. Such migration or spawning can be done in the control infrastructure based on global application states monitoring and synchronizers.

The other method can be called “non-migrative”. It assumes the we do not migrate activities among computing resources on the Grid but we build the activities in this way that they are able to transfer parts of their tasks to other activities which are in this sense underloaded. Such idea of load balancing in scientific computational workflows requires a special instrumentation of workflows – creating structures of communicating activities, which can internally observe the efficiency of their execution and can regulate the amont of work they have to execute without changing the structure of the workflow. Such a method of control based on adaptive activities can be easily implemented through monitoring of global activity internal states by Grid-level synchronizers. It involves evaluation of control predicates on such states and can cause dynamic displacement of work among the activities.

The paper presents how the control based on monitoring of global application states can be used to co-ordinate dynamic scientific computations workflow execution including the load balancing of activities execution at workflow run-time. The proposed ideas extend the ordinary task and application workflow schemes. Implementation of such paradigms based on local and global synchronizers will be discussed, using graph representations of workflows, to illustrate the mechanisms for dynamic changes of workflow element functionality and efficiency of their execution, which can be applied in time optimization of Grid scientific applications.

References

1. M. Tudruj, J. Borkowski, D. Kopanski, Graphical Design of Parallel Programs with Control Based on Global Application States Using an Extended P-GRADE System, in Distr. and Parallel Systems, Cluster and GRID Comp., Kluver, Vol. 777, 2004.
2. D. Kopanski, J. Borkowski, M. Tudruj, Co-ordination of Parallel Grid Applications Using Synchronizers, Proc. of PARELEC 2004, Dresden, Germany, pp 323-327
3. M. Reichert, P. Dadam, A Framework for Dynamic Changes in Workflow Management Systems, in Proc. 8th Int. Workshop on database and Expert Systems Applications, IEEE 1997, pp 42-48
4. M. Kwak, D. Han, J. Shim, A Framework Supporting Dynamic Workflow Interoperation and Enterprise Application Integration, Proc. of 35th Hawaii Int. Conf. on System Sciences, IEEE 2002