IST-2001-32133
GridLab - A Grid Application Toolkit and Testbed

Use Cases for Adaptive Components

Author(s):	Thilo Kielmann, Gabrielle Allen, Matthew Shields, Ian Taylor, André Merzky, Jarek Nabrzyski
Document Filename:	GridLab-7-UCR-0001-UseCasesReport
Work package:	WP7: Adaptive Grid Components
Partner(s):	VU, AEI (for WP1 and WP2), UWC (for WP3), ZIB (for WP8), PSNC (for WP9)
Lead Partner:	Vrije Universiteit (VU)
Config ID:	GridLab-7-UCR-0001-2.0
Document classification:	IST

Abstract: This report describes a set of use cases for adaptive components to be addressed in the course of the GridLab project. The use cases have been determined by the needs of the partner work packages mentioned above.

Changes since Version 1.0 of this report:
The selection of use cases is driven by those work packages for which adaptive components are built. According to changing requirements of these work packages, the set of use cases has been modified accordingly. The new set of use cases has been streamlined to address all important use cases while avoiding redundancy between them. This way, the needs of WP7's client work packages can be served optimally.

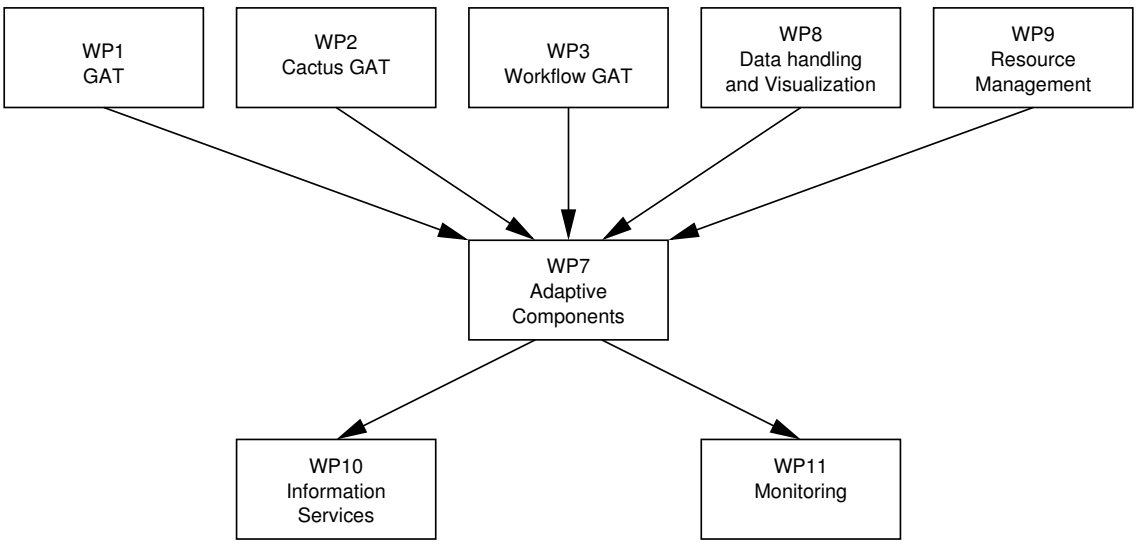
Contents

1 Introduction

Grid resources are highly dynamic: network connections have varying bandwidth and latency, available CPU resources come and go. Static (inflexible) strategies and implementations are thus not suitable in Grid environments. Adapting application behavior (together with middleware layers) thus is inevitable.
In WP7, we will generalize the basic techniques of gathering relevant monitoring data, correlating this data with performance models (specific to toolkit modules), and short-term prediction of application performance, finally leading to behavior adaptation. The WP will develop generic components for implementing adaptive behavior.
The goal of this work package is to provide adaptive application components to other modules of the application toolkit. Common to these components is that they take dynamic information about Grid resources (network status, CPU availability, memory footprint, etc.) as input to module-specific performance-prediction models and implement adaptive strategies that let applications efficiently use the given resources. For grid-aware applications, this general mechanism is used for numerous purposes like data access, task scheduling, and dynamic application reconfiguration. Although use-case specific APIs and event models are necessary, the basic techniques are similar. The objective is to develop adaptation components that will be instantiated to their use from other modules of the GAT and from within other GridLab services. For the input data, this WP will rely on the work performed in WP10 and WP11.

2 Relation to other work packages

The following simple diagram depicts the relation of WP7 with other work packages in GridLab. The diagram shows a is-client-of relation. Work packages 1, 2, 3, 8, and 9 thus are clients of WP7; adaptation components will be built for these work packages' needs. We call these the *client work packages* of WP7. WP7 is a client of work packages 10 and 11. Work package 11 provides the infrastructure for providing actual monitoring data while WP10 provides meta data about the monitoring system like contact information and metrics descriptions.



Technically speaking, the adaptation components will become plugins to the GAT modules (adaptors and services) of the client WPs of WP7. Other parts of the GAT, or even applications, will not observe their presence directly.

2.1 Relation between WP7 and WP9

WP7 supports WP9 by providing adaptation components for the purposes of the GRMS resource management system built by WP9. Besides that, both work packages complement each other in their functionality for the overall project goals. This can be summarized as follows:

- WP9 is responsible for resource management in general. As such, it builds a resource broker and an application scheduler, forming substantial parts of an overall *GridLab Resource Management System* (GRMS). All allocations (assignments, scheduling) of grid resources to applications are supposed to be handled by the GRMS.
- WP7 is responsible for application-level optimization in order to efficiently exploit resources by an application, after the resources have been allocated by the GRMS. This may involve placement and scheduling of application subtasks within the GRMS-allocated resources, as for use case 3.1.
- An adaptation component (as built by WP7) may indicate the necessity for additional resources during an application run. In this case, such additional resources will be requested via the GRMS. Application migration is one example of such resource allocation at runtime.

3 Use Cases

In the following, we list the individual use cases for adaptation components to be addressed throughout the GridLab project. The primary selection criterion for the use cases is the need for adaptive behavior from the client work packages. Of secondary interest is to achieve a coverage of possible resources (like network bandwidth, CPU speed, memory, I/O throughput), the changing availability of which causes the need for adaptation.
For each use case, we describe the scenario and what can be gained by adaptation to certain resources. We list the respective client work packages and the resources to be dealt with. We also outline the basic functionality of an adaptation component for the use case. However, we do not fix the details of an API between the adaptation component and its clients here, as this becomes part of the actual deployment of the use cases.

3.1 Resource Selection for Parallel Triana Units

Scenario: Triana programs are formed by so-called *units*, connected to a flow graph, through which data items flow from sources to sinks. Each unit performs some computation to process the data. Units may have multiple inputs and outputs, and they may be composed from other (simpler) units. Units can be parallelized in a transparent way. An input stream of a parallel unit can be split into blocks, which can be handled independently in parallel by multiple instantiations of a sequential unit. When the input blocks have been processed, the resulting outputs are combined again into a single output stream.
In a Grid environment, a Triana flow graph is supposed to run on a distributed set of machines. Parallel units need to be mapped onto a set of those machines for execution. The problem is to assign the units to the machines such that the overall system computes as fast as possible (with maximal throughput). Care needs to be taken of machines having different computing speeds and units having different computational needs.

Client WPs: WP3 (Workflow GAT).

Resources: CPU speeds of individual machines, network bandwidth.

Basic Functionality: The estimation of unit execution time can be based on empirical measurements. This may happen dynamically during the actual run of the Triana application.
The adaptation component should be implemented as a Triana unit that computes the optimal assignment of parallel units to machines. Whenever the mapping of units to machines is supposed to change, this second unit can be called again to compute a new mapping. Such mapping changes result from changes in the application's problem size, changing network performance or from the availability of additional machines, or the removal of machines from a running application.

3.2 Transfer Protocol Optimization

Scenario: Migration of a parallel (e.g., Cactus) run from one machine to another machine. Consider that the application needs a large input data set that must be transfered from the source to the destination machine. The data should be transfered as efficiently as possible. GridFTP (and other transfer protocols) performance depends heavily on the caracteristic of the network link, such as latency and available bandwidth, and on the tranfer parameters that are used. Therefore, these parameters (e.g., send and receive buffer sizes, the number of parallel streams) should be carefully tuned for each seperate link to achieve optimal performance.

Client WPs: WP2 (Cactus GAT), WP8 (Data Management), WP9 (Resource Management).

Resources: network bandwidth, network parameters, network topology.

Basic Functionality: The adaptive components framework provides bandwidth and latency estimates, network topology data and information on network parameters such as the maximum send and receive buffers that can be used on a particular system.
The bandwidth estimates are correlated to the throughput achieved by the transfer protocol that is used. The adaptive component uses heuristics to estimate the best options for the available transfer method. Tunable options include send and receive buffer size, and the number of parallel TCP streams that should be used. These parameters can depend on the network topology, latency, capacity and the available bandwidth.
Given a prediction for the bandwidth and the optimal transfer parameters, transfer time estimations can be computed. From multiple alternatives, the fastest one can be chosen.
The migration involves the redistribution of application data between the compute nodes. This necessary redistribution can happen before, during, or after the transfer over the network, allowing to minimize transfer time by choosing the strategy that makes the most efficient use of the network.
An adaptation component can be passive and be called from the process that triggers the migration. The component returns a choice between the possible alternatives, depending on which level (from target machine to application-redistribution strategy) the selection is to be made.
In the case of estimating transfer time for starting migration in advance (before the compute time on the current machine ends), the adaptation component needs to be called from a separate watchdog process that monitors the progress of the parallel job, including compute time, available memory and disk space. Still, the estimating adaptation component remains passive.

3.3 Replica Selection

Scenario: Migration of a parallel (e.g., Cactus) run from one machine to another machine. Consider that the application needs a large input data set that has been replicated several times throughout the grid. The time needed to transfer the application input data from a replica to the target machine needs to be estimated in advance, for possibly multiple purposes. First, a migration target machine may be selected by the time it takes to migrate to it (in behalf of the GRMS). Second, after a migration target has been chosen, one of possibly many replicas must be selected. Third, after target machine and source replica have been selected, the transfer software/protocol may be optimized (like GridFTP parameter tuning). Finally, an estimation of data transfer time may be necessary to start migration in time, before the allocated compute time on the migration source machine ends.

Client WPs: WP2 (Cactus GAT), WP8 (Data Management), WP9 (Resource Management).

Resources: network bandwidth, network parameters, network topology.

Basic Functionality: The adaptive components framework provides bandwidth estimates and network topology information. The bandwidth estimates are correlated to the throughput achieved by the available transfer protocols. Transfer time estimations can be computed from this. From multiple alternatives, the fastest one can be chosen. As *alternatives*, the choice might be between multiple replica locations and between multiple transfer protocols.
The migration involves the redistribution of application data between the compute nodes. This necessary redistribution can happen before, during, or after the transfer over the network, allowing to minimize transfer time by choosing the strategy that makes the most efficient use of the network.
An adaptation component can be passive and be called from the process that triggers the migration. The component returns a choice between the possible alternatives, depending on which level (from target machine to application-redistribution strategy) the selection is to be made.
In the case of estimating transfer time for starting migration in advance (before the compute time on the current machine ends), the adaptation component needs to be called from a separate watchdog process that monitors the progress of the parallel job, including compute time, available memory and disk space. Still, the estimating adaptation component remains passive.

3.4 Remote Data Visualization

Scenario: A user wants to visualize the output of a completed, or even a running, remote computation (like a Cactus run). The critical resource is the network bandwidth that limits either the quality or the delay of the visualization, or even both.
The user needs to determine a desired tradeoff between quality and delay. For this purpose, the user specifies the maximal useful resolution of the image, and a desired frame rate. The remote visualization software can then provide the best possible images within the constraints of the user and the technical possibilities.

Client WPs: WP8 (Data Management).

Resources: network bandwidth.

Basic Functionality: The key to this adaptation problem is to perform visualization using *progressive resolution*. The visualization software can construct the image hierarchically, starting with a very low resolution, up to the maximum quality that can be achieved within the constraints of frame rate and network bandwidth. The resolution may even vary between different parts of an image.
The adaptation component uses its network bandwidth data from the monitoring system, produces short-term predictions, and correlates this with transfer times of the needed data over the measured link. The adaptation component can then provide an estimate for the data volume that can be sent within a given time limit. This estimate can be used by the visualization software to control the generated image resolution, helping to avoid starting with very low resolutions if it can be predicted that a better resolution will be possible according to the network conditions.

3.5 Queue Waiting Time Estimation

Scenario: The GRMS is supposed to schedule an application request such that the application starts running as soon as possible.
For this purpose, the scheduler needs information about predicted wait time in processor queues when scheduled to a given machine, or a combination of multiple machines, the latter in case of co-allocation. Also, the CPU and memory load of the machines with a queuing system has to be taken into account.

Client WPs: WP9 (Resource Management).

Resources: CPU speed, processor queue status, CPU and memory load.

Basic Functionality: In this scenario, the adaptation component focuses on performance prediction for given time series of processor queue waiting times and of historical data for application run times. The application performance model needs to predict application runtime for a given application, a given problem data classification, and given machine architecture.