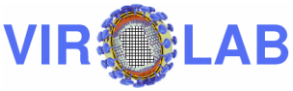
Master of Science Thesis

Optimization of Grid Application Execution

Joanna Kocot, Iwona Ryszka

supervisor: Marian Bubak, PhD

advice: Maciej Malawski, MSc



Outline

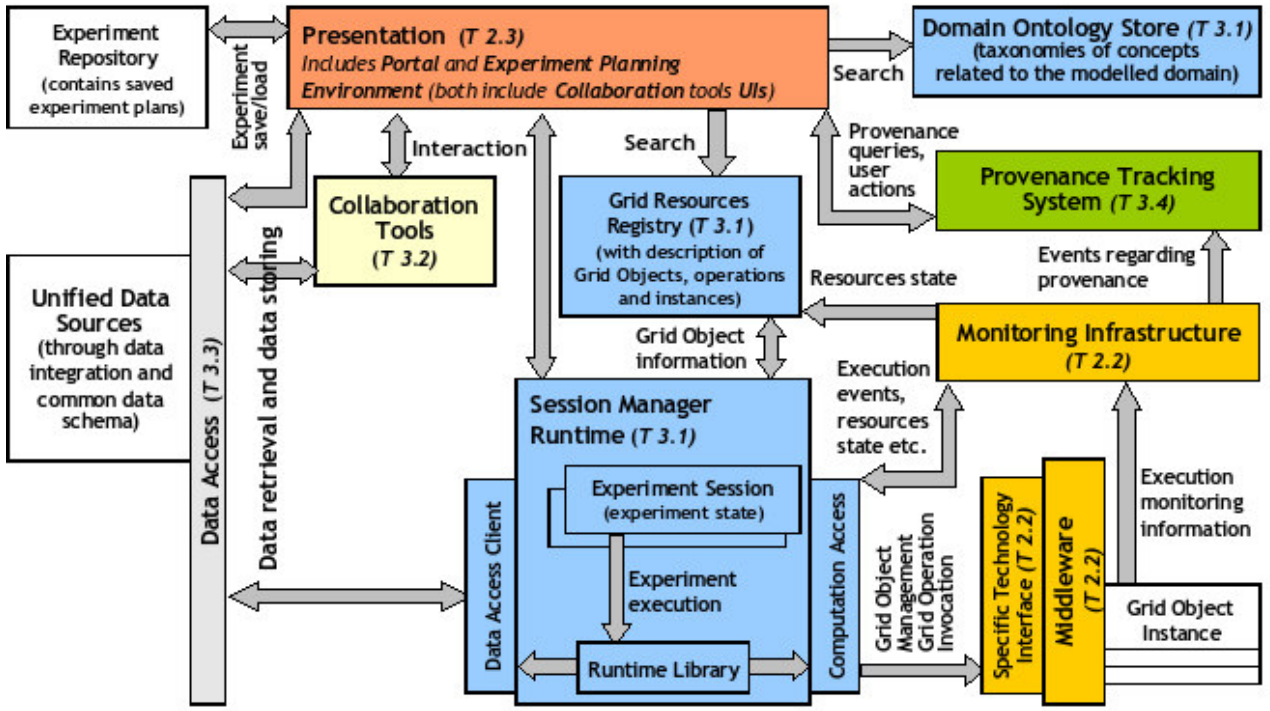
- MSc Goals
- ViroLab Environment
- Optimization Model
- Optimizer Architecture
- Optimizer Implementation
- Optimizer Testing
- Summary

MSc Thesis Goals

- Providing a Virtual Laboratory subsystem for optimization of Grid-based applications
  - Identification of available optimization solutions in Grid computing
    - Research into related work to gain a wider view on the problem and find solutions useful for the thesis.
  - Identification and analysis of the problem of optimization in ViroLab
    - Problem statement taking into account the target environment.
  - ViroLab Optimizer design and development
  - Proving the usefulness of the developed Optimizer for ViroLab
    - Execution of unit tests, integration tests and quality tests.

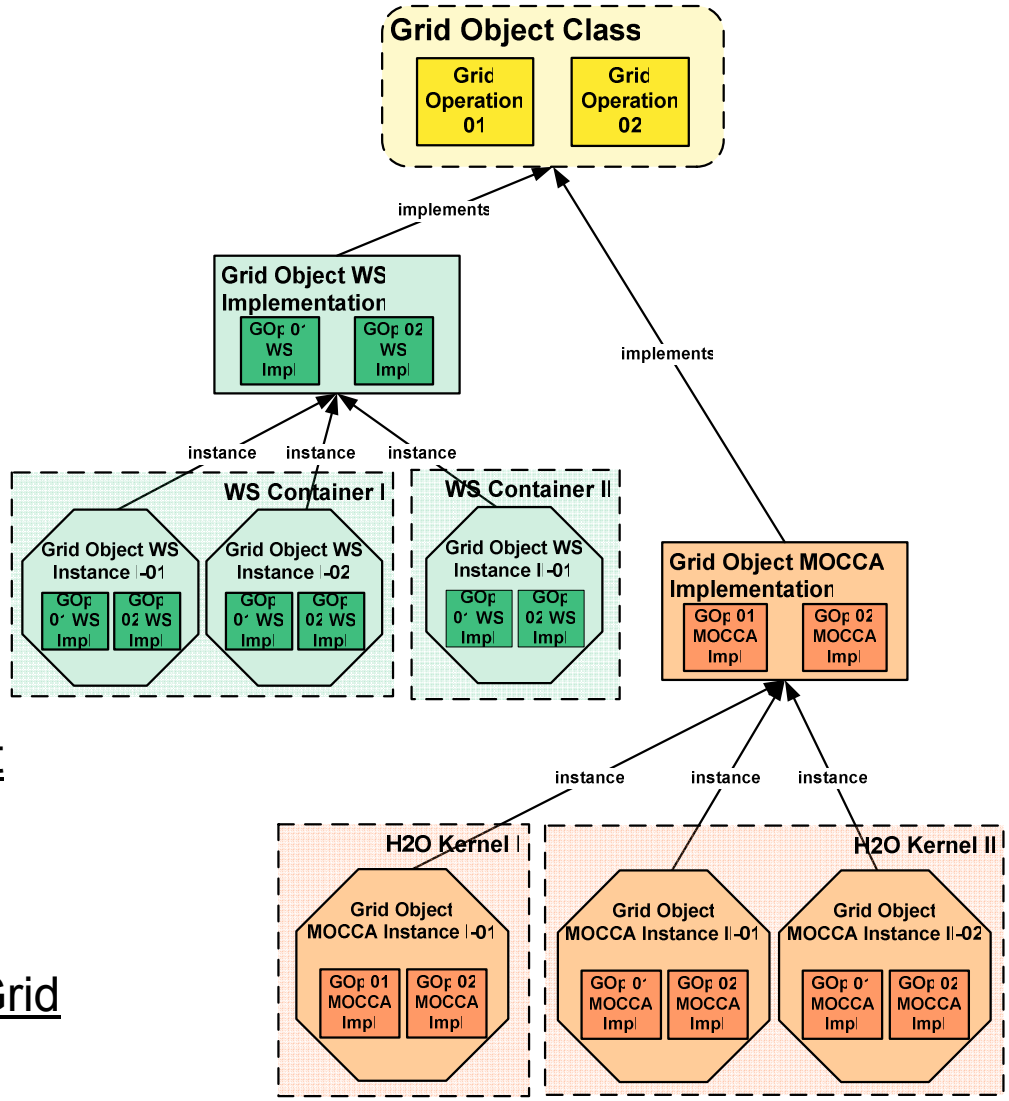
ViroLab – Virtual Laboratory

- A research project of the EU 6th Framework Program
  - Its mission is to provide researchers and medical doctors with a virtual laboratory for infectious diseases (mainly HIV virus infections).
- ACK Cyfronet AGH responsible for development of ViroLab Virtual Laboratory Runtime
  - Runtime for execution of experiments.
  - Developed with use of Grid infrastructure and heterogeneous resources.



Levels of Abstraction – ViroLab Entities

- ViroLab Experiment
  - Composed of calls to Grid Operations
- Grid Object Class
  - Interface declaring Grid Operations
  - Can be implemented by various Grid Object Implementations
- Grid Object Implementation
  - Static entity - codebase
  - Represented by Grid Object Instances
- Grid Object Instance
  - Created by deploying Grid Object Implementation on Grid Resource



Motivation for Optimization in ViroLab

- While executing an experiment, the ViroLab Runtime:
  - Knows which Grid Object Class is able to perform a certain operation.
  - Needs information which instance of the Grid Object Class (Grid Object Instance) should perform the operation.
- The aim of ViroLab Optimizer is to decide:
  - Which Grid Object Implementation will be the most suitable to perform the processing.
  - Which ready Grid Object Instance of this Grid Object Implementation will be the most suitable to perform the processing.
  - Whether the Grid Object Instance should be chosen or a new one is to be deployed.
  - Where (on which Grid Resource) a new Grid Object Instance should be created.
- Optimization result (solution): Grid Object Instance or Grid Object Implementation + resource URL

Optimization Model

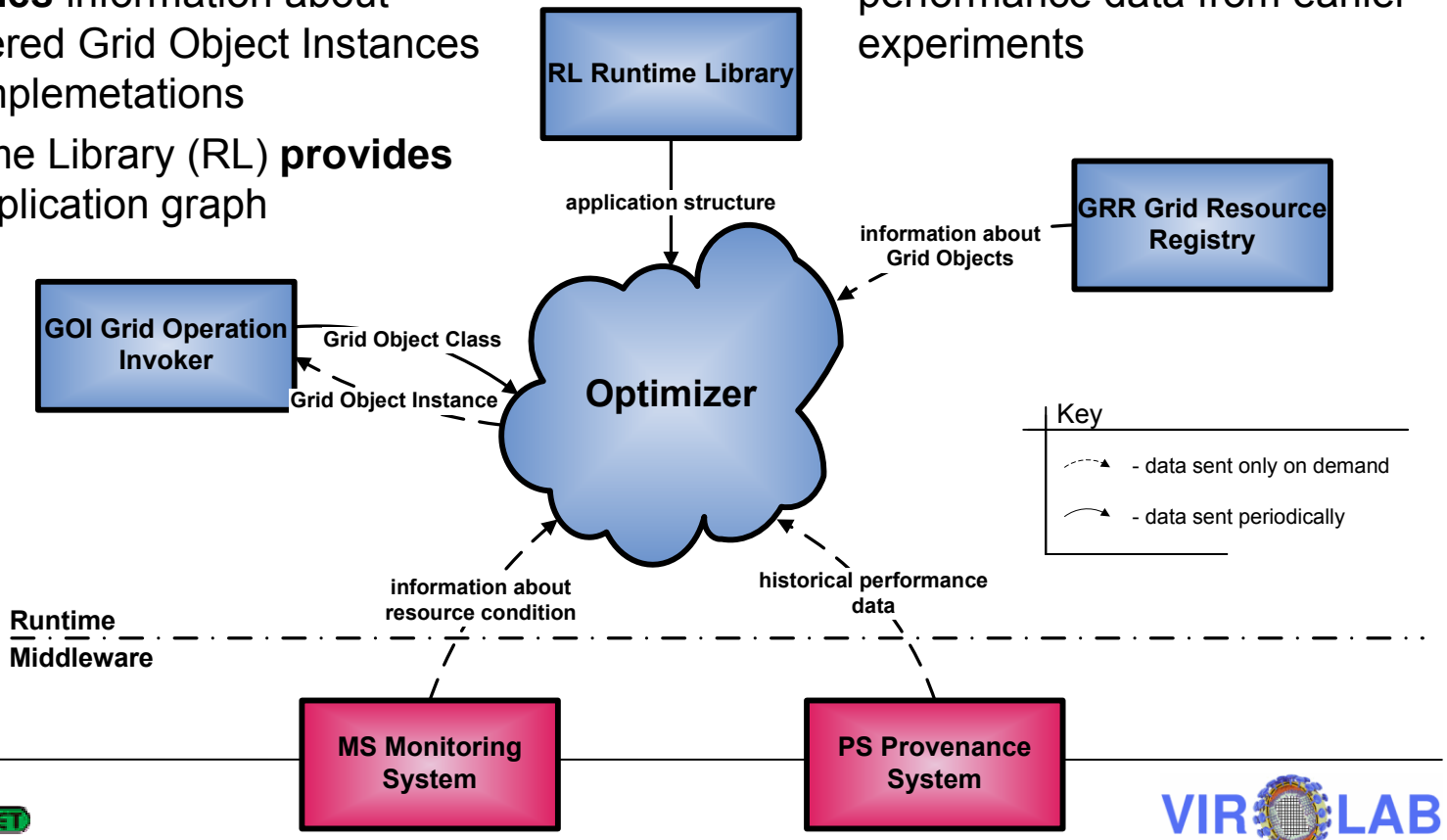
- Characteristics of the ViroLab Optimizer
  - *No direct control over resources* – works like a broker or an agent.
  - *No exclusive access to resources* – reliability of optimization information is not as high as it would be when obtained from a local scheduler.
  - *No queue* – no management of jobs after their submission.
  - *Global* – one optimizer with a system-wide performance objective.
  - *Hybrid solution between static and dynamic optimization* – both historical data and information, if available at runtime, are used.
  - *Application centric* – optimization process concentrates on the performance of application.
  - *Adaptive* – the optimization process can be dynamically adapted to changes in the ViroLab environment.

Optimization Modes

- Available optimization modes:
  - *short-sighted optimization mode*
    - The aim is to choose an optimum solution only for one Grid Object Class at a time.
  - *medium-sighted optimization mode*
    - Finds solutions for a group of Grid Object Classes at a time.
    - Tasks are not reordered nor arranged in queues.
  - *far-sighted optimization mode*
    - Similar to the above mode.
    - The whole application is being analyzed at a time.
    - Ordering the Grid Object Classes is performed by taking into account dependencies between them.

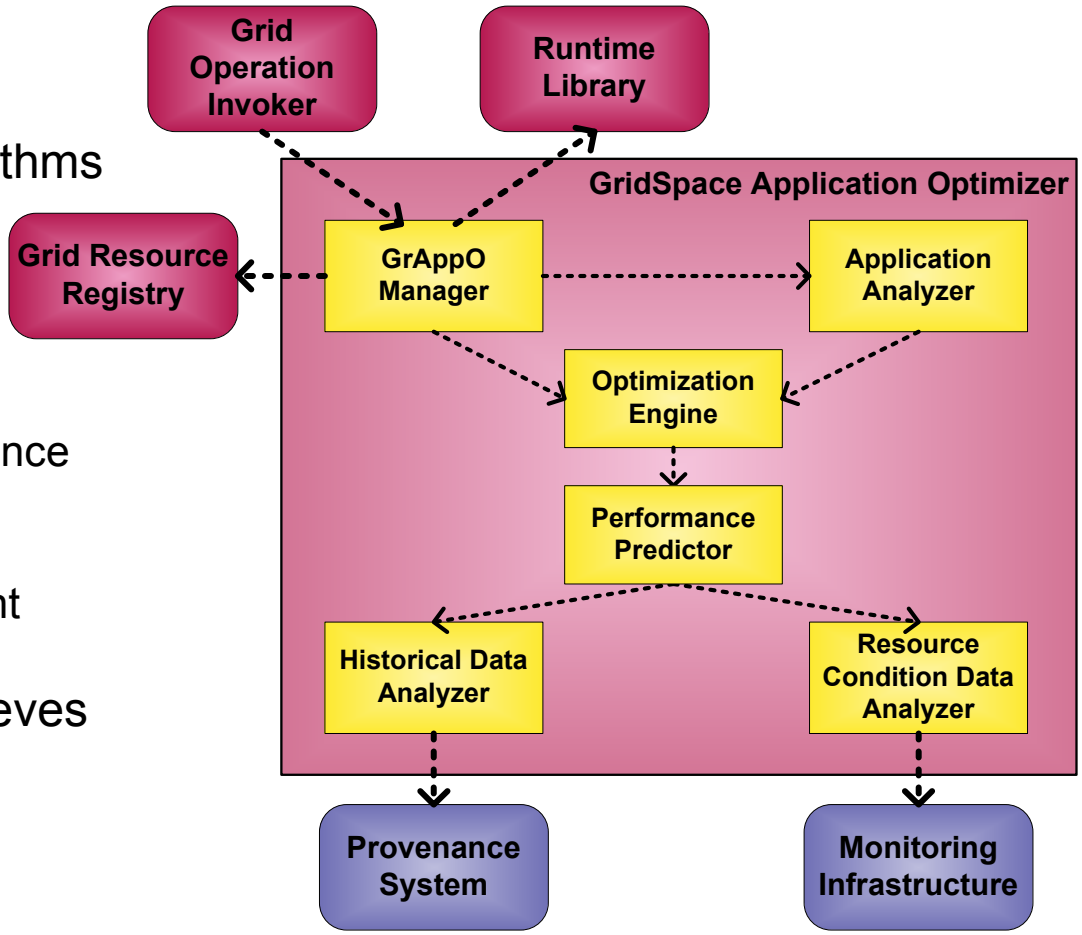
Cooperation with other ViroLab Components

- Runtime
  - Grid Operation Invoker (GOI) **queries** for optimum Grid Object Instance or Implementation
  - Grid Resource Registry (GRR) **provides** information about registered Grid Object Instances and Implemetations
  - Runtime Library (RL) **provides** the application graph
- Middleware
  - Monitoring Infrastructure **provides** resources condition information
  - Provenance System **provides** performance data from earlier experiments

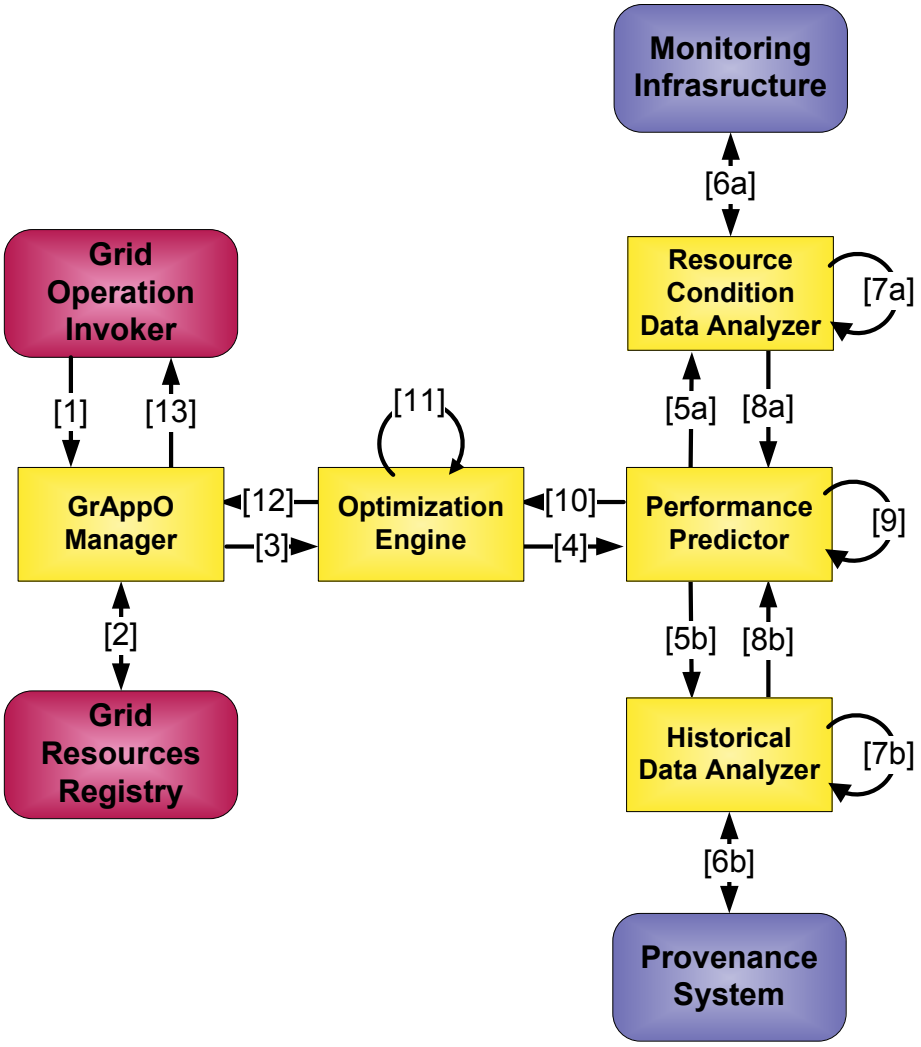


General Architecture of GridSpace Application Optimizer (GrAppO)

- **GrAppO Manager** – coordinates GrAppO components
- **Optimization Engine** – calculates optimization algorithms
- **Performance Predictor** – estimates performance of possible solutions using:
  - **Historical Data Analyzer** – analyzes historical performance data
  - **Resource Condition Data Analyzer** – analyzes current state of resources
- **Application Analyzer** - retrieves the application graph and analyzes it



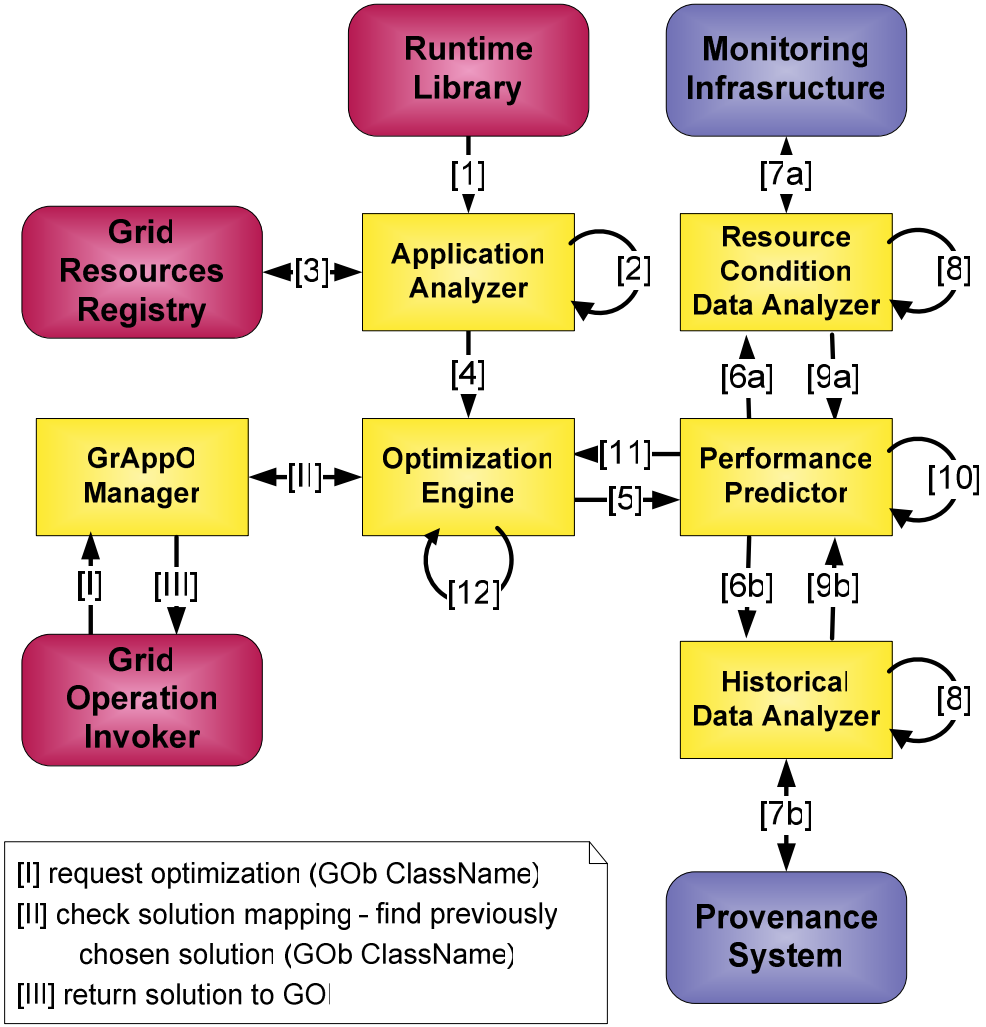
Control Flow in GrAppO: Short- and Medium-Sighted Optimization



[1] request optimization (GOb ClassName(-s)*)
[2] get GOb Instance, Implementation
and resource information (GOb ClassName(-s))
[3] request search for optimum solution
(information from GRR)
[4] request performance estimation
(information from GRR)
[5a] check resources condition (resource locations)
[5b] check historical performance data
(GOb Implementations, resource locations)
[6a] query the Monitoring Infrastructure (locations)
[6b] query the Provenance System
(GOb Implementations, resource locations)
[7a] analyze resource condition data
[7b] analyze historical performance data
[8a, 8b] return results of the analysis
[9] estimate performance - for all possibilities
[10] return estimation results
[11] evaluate scheduling algorithms to find best solution(-s)
[12] return the result: GOb Instance ID(-s)
or GOb Impl(-s) + resource location(-s)
[13] forward the obtained solution to GOI

* the -s form is used in medium-sighted optimization

Control Flow in GrAppO: Far-Sighted Optimization



[I] request optimization (GOb ClassName)
[II] check solution mapping - find previously chosen solution (GOb ClassName)
[III] return solution to GOI

[1*] request optimization (application)
[2] process the application
[3*] get GOb Instance, Implementation and resource information (GOb Classes) - about classes included in application
[4] request search for optimum solution (information from GRR)
[5] request performance estimation (information from GRR)
[6a] check resources condition (resource locations)
[6b] check historical performance data (GOb Implementations, resource locations)
[7a] query the Monitoring Infrastructure (locations)
[7b] query the Provenance System (GOb Implementations, resource locations)
[8a] analyze resource condition data
[8b] analyze historical performance data
[9a, 9b] return results of the analysis
[10] estimate performance - for all possibilities
[11] return estimation results
[12] map solutions to GOb Classes

* the contact with RuntimeLibrary and GRR is realized through GrAppO Manger

GrAppO Implementation

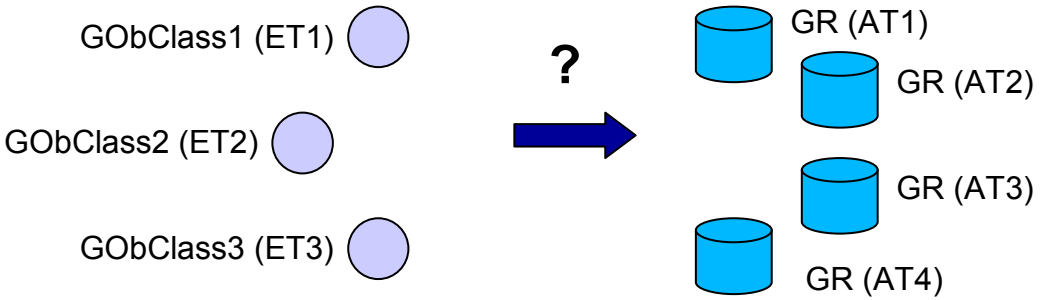
- Current status
  - Short- and medium- sighted optimization mode.
  - Possible analysis of information from all data sources.
  - Connection to Grid Resource Registry (other data sources unavailable).
- Adaptive optimization using XML-based Optimization Policy
  - Determines optimization algorithms.
  - Declares preferred implementation type (e.g. Web Service).
  - Specifies additional data sources.
- Technologies:
  - Core of GrAppO: Java 2 Platform SE 5.0
  - Connection to GRR service: Codehaus XFire – Java SOAP framework
  - GrAppO unit tests: JUnit – testing framework

GrAppO Testing

- Unit tests
  - All main classes of GridSpace Application Optimizer are covered.
- Integration tests
  - Testing GrAppO integration with Grid Resource Registry and Grid Operation Invoker – communication channels work correctly.
  - Monitoring System and Provenance System Tracking are not available yet, but in GrAppO the required interfaces are ready.
- Acceptance tests
  - Successful execution of real ViroLab experiments (weka, alignment, subtyping, from-geno-to-drug resistance).
  - Performed within a distribution of ViroLab Runtime – in the target environment (available at http://virolab.cyfronet.pl).

Quality tests of GrAppO (1) - Introduction

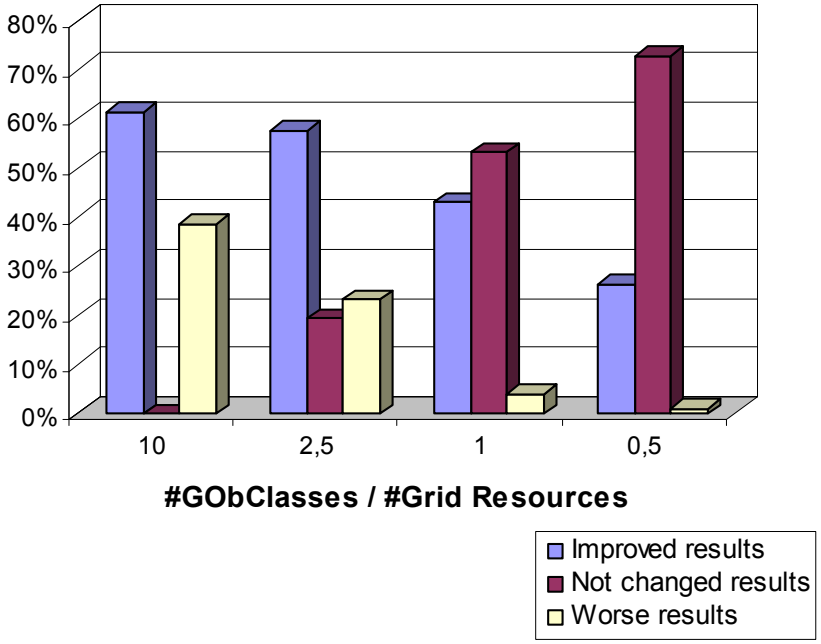
- Performed in a simulated environment
  - Monitoring Systems and Provenance Tracking systems were implemented as mock components providing random data.
- Metrics: *Minimum Completion Time (MCT)*
  - Completion Time – a moment of time when a resource completes a Grid Object Class's operation: after finishing execution of previously planned jobs (***AT*** – *Availability Time*) and executing the operation (***ET*** – *Execution Time*)



- Optimization objective: minimization of *makespan* (maximum of MCTs of Grid Object Classes from a given set)
- Used heuristics
  - *Min-min* - considers the MCT of each Grid Object Class (average of its operations) on available Grid Resources and chooses the one with the lowest MCT
  - *Max-min* - again the MCT for each Grid Object Class is evaluated. The one with the *maximum* MCT is assigned to the corresponding Grid Resource.

Quality tests of GrAppO (2) – Comparison of Optimization Modes

- Improvement of makespan while using medium-sighted optimization mode in comparison to short-sighted optimization mode – for different proportions of Grid Object Classes to available Grid Resources
  - Percentage of improved / not changed makespans



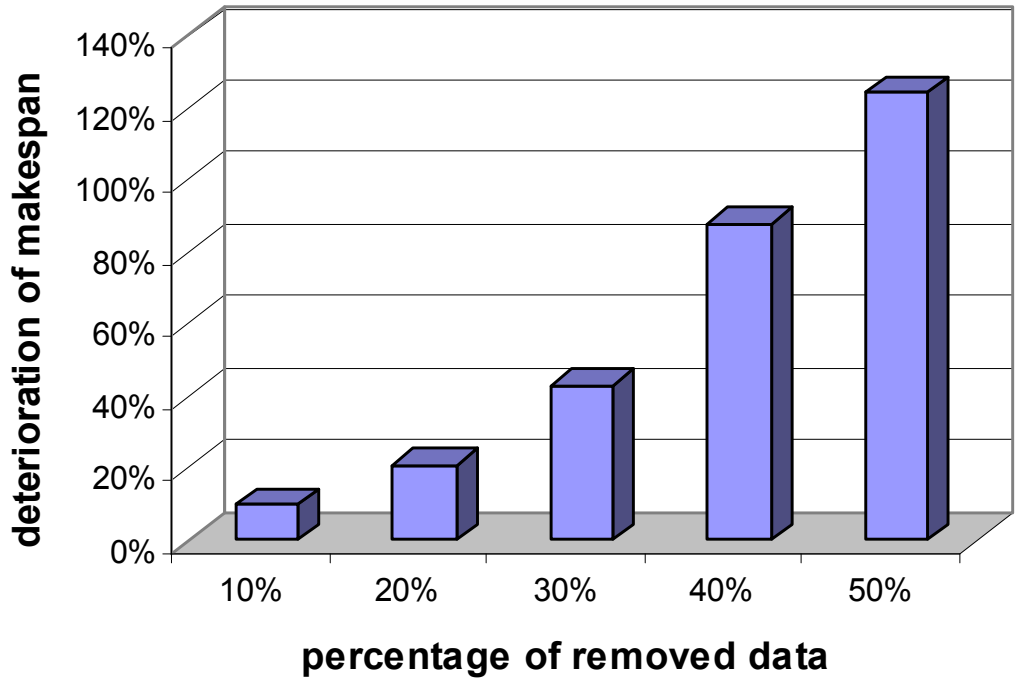
  - Average improvement of makespan

Quality tests of GrAppO (3) – Comparison of Optimization Algorithms

- If no information about resources is provided, a random solution is chosen.
- Every tested optimization algorithm brings over 200% better result than choosing random solution – even in short-sighted optimization mode.
- The tested heuristics (Min-min and Max-min) give similar results
  - Max-min heuristic is better when some of the Grid Object Classes to optimize has significantly longer execution time (ET) than others.
  - Improvement of 5.6% in comparison to Min-min heuristic.

Quality tests of GrAppO (4) – Influence of Information Quality

- The optimizer is easily influenced by the quantity and the quality of information gathered from external data sources.



Summary

- The main goal of the thesis – providing an optimizer for ViroLab was successfully achieved.
- GrAppO was integrated with ViroLab and operates for real experiments correctly.
- Executed tests gave satisfactory results and proved the benefits of introduction different optimization modes and algorithms.
- Future work:
  - Implementation of real connections to other ViroLab components – Monitoring System and Provenance Tracking System.
  - Implementation of far-sighted optimization mode.
  - Graphical interface for GrAppO configuration.

For more information please visit:

http://www.virolab.org

http://virolab.cyfronet.pl

http://gforge.cyfronet.pl/projects/grappo