Composing Distributed Services for Selection and Retrieval of Event Data in the ATLAS Experiment

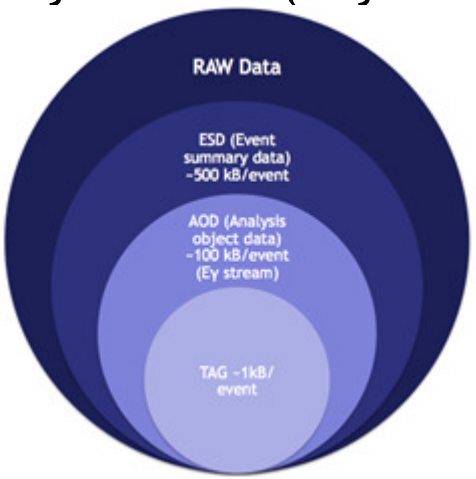
Elisabeth Vinek, CERN & University of Vienna
for the ATLAS Collaboration

Outline

- TAGs: ATLAS event-level metadata
  - What is it?
  - Services
- Service selection challenge:
  - System requirements
  - QoS based service selection
  - Steps of the adopted approach
  - Description of system components
  - Important system metrics
  - Logging infrastructure
  - Optimization objectives
  - Simulation and resource planning
- Expected Results
- Conclusions

TAGs: ATLAS event-level metadata

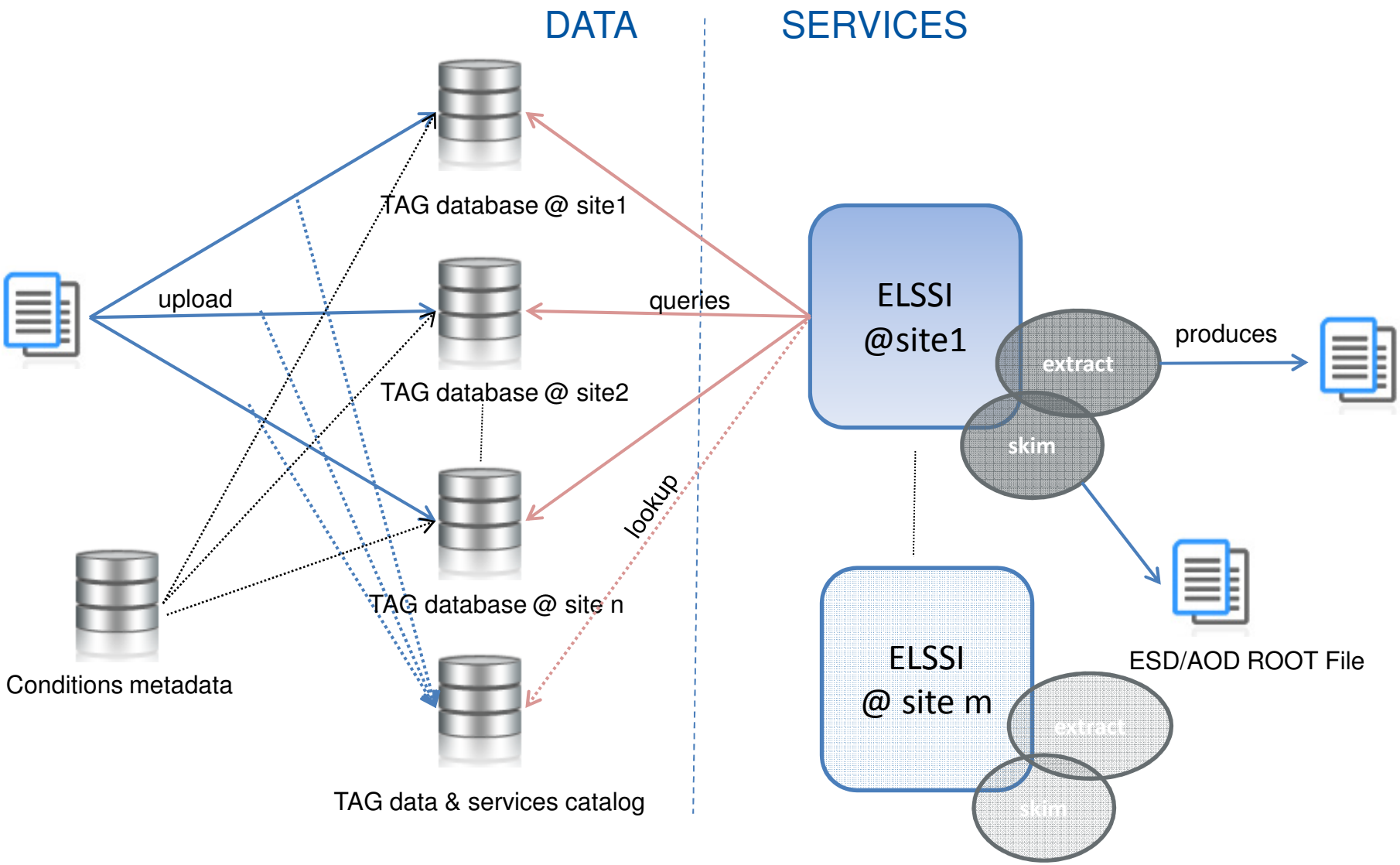
- TAGs are event-by-event metadata records containing:
  - key quantities that identify and describe the event, intended to be useful for event selection, and
  - sufficient navigational information to allow access to the event data at all prior processing stages: RAW, ESD, and AOD
- TAG is not an acronym
- Contains more than 200 variables for each event to make event selection easy and efficient
- Variables decided by 2006 Task force and maintained by the PAT (Physics Analysis Tools) group
- TAG contents are quite stable since then
- Data representation evolves only for easier use
  - (e.g. Trigger decision at three levels)



TAG data sources and services

- **File based TAGs** are built from AOD content and written into files when AOD files are merged.
- **Relational TAGs** are uploaded to Oracle databases using POOL Collection utilities.
- Main **TAG services**, ELSSI service suite:
  - Relational databases serving as data sources
  - iELSSI: Event Level Service Selection Interface
  - Extract: extracting a TAG ROOT file based on a database selection
  - Skim: skimming events based on a database selection, returning an AOD or ESD file
  - Some parts of the suite implementing specific functionality can be seen and distributed as single services, e.g. trigger decoding.
- Sites (Tier-0, some Tier-1s and Tier-2s) are hosting TAG data and services on a volontary basis.
- Data as well as services are thus **distributed** across ATLAS sites.

TAG System Overview



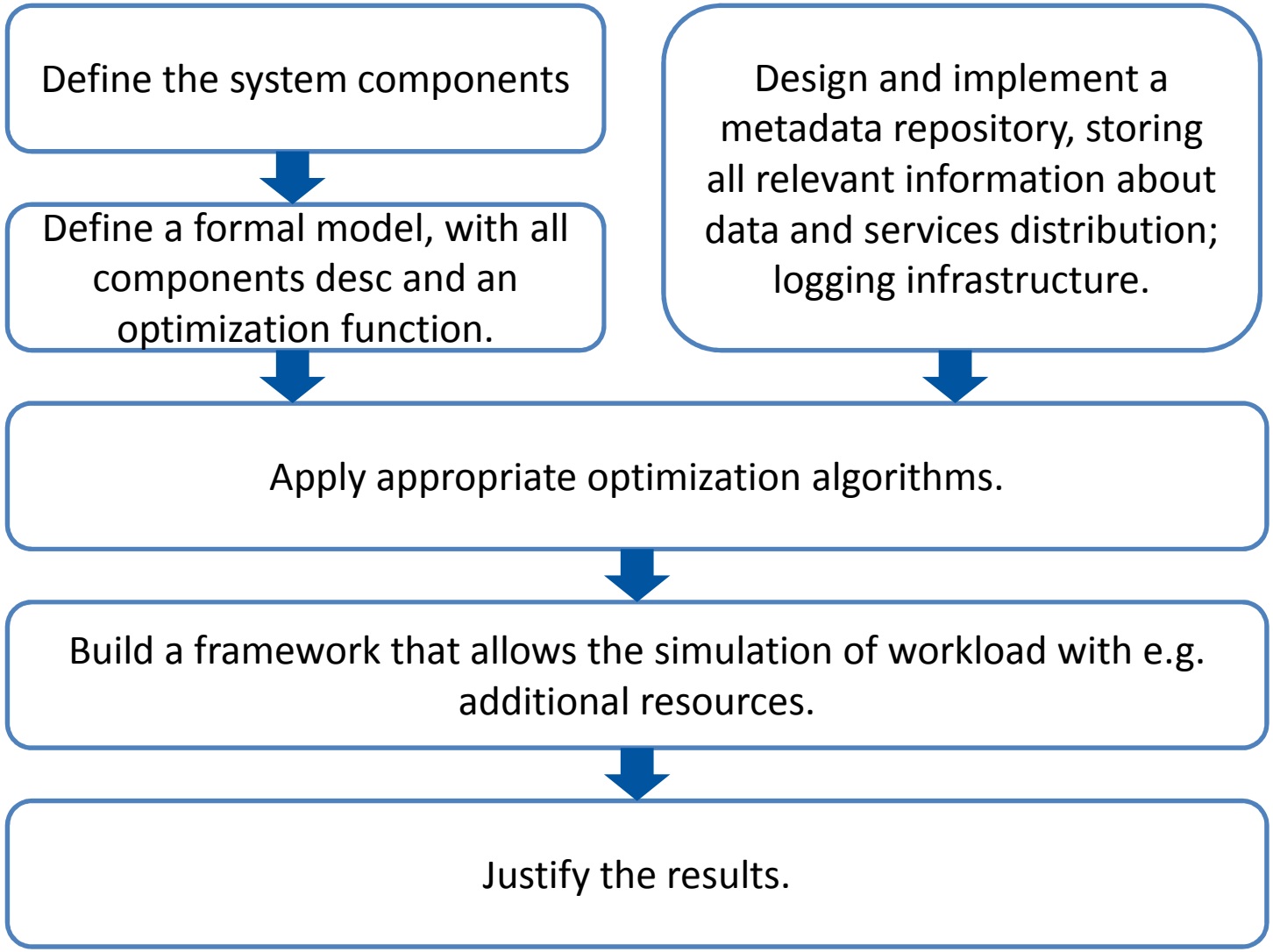
Requirements and Challenge

- Requirements as defined from the system‘s provider point-of-view:
  - 1 - make all TAG databases look like one, i.e. make the data distribution transparent to the user.
  - 2 - make the service deployments transparent to the user.
  - 3 - make an efficient use of all available resources and services.
  - 4 - enable load balancing and fail over mechanisms for TAG resources/services (-> build a reliable system) and serve as many users as required with the given resources.
- Additional requirements as (implicitly) defined by the users:
  - 5 - Get the *fastest* possible – and correct – answer for TAG requests.
  - 6 - Support for several use cases (individual requests, batch jobs).
- The challenge is to automatically decide:
  - which services are needed for a specific request
  - and which replica of each service should be selected,
  - in order to best meet the above requirements.

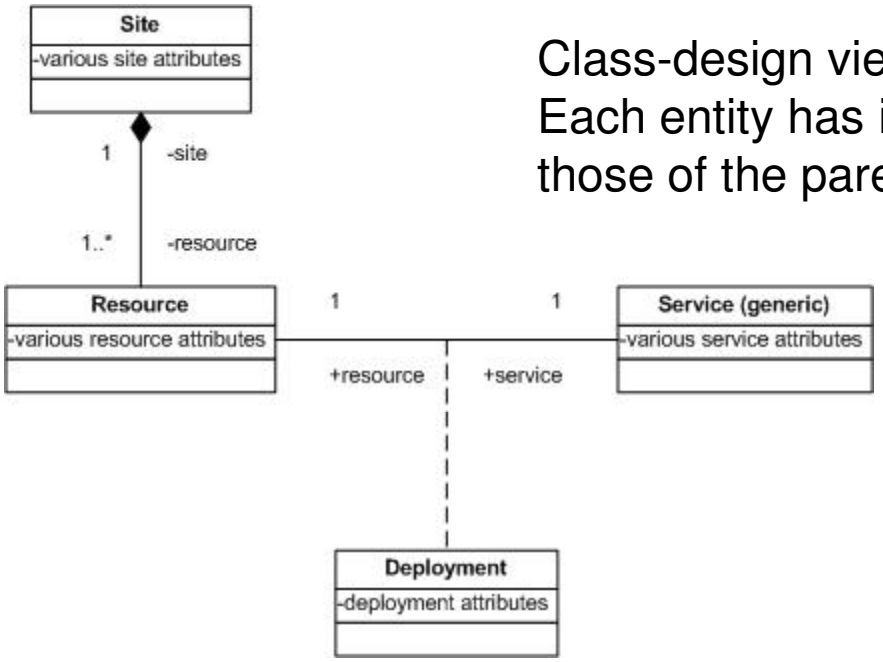
QoS-based service selection

- QoS (Quality of Service) – based service selection is a wide-studied research area.
  - However, most of the approaches are motivated by business use cases where end-user costs are the main driver.
  - Additionally, selection is mainly done on services that are outside the domain controlled by the selection mechanisms, i.e. the system‘s characteristics are less known than in our case, and trust is an important part.
- Related areas:
  - Site selection in Grid systems => more about scheduling.
  - QoS routing, e.g. load-balancing between web servers => homogeneous infrastructure, less relevant attributes to consider.
- A detailed and formal description of the system is the starting point for efficient service selection.
- In our particular case, **all** system components are **well known**, and the system can be analyzed in detail.

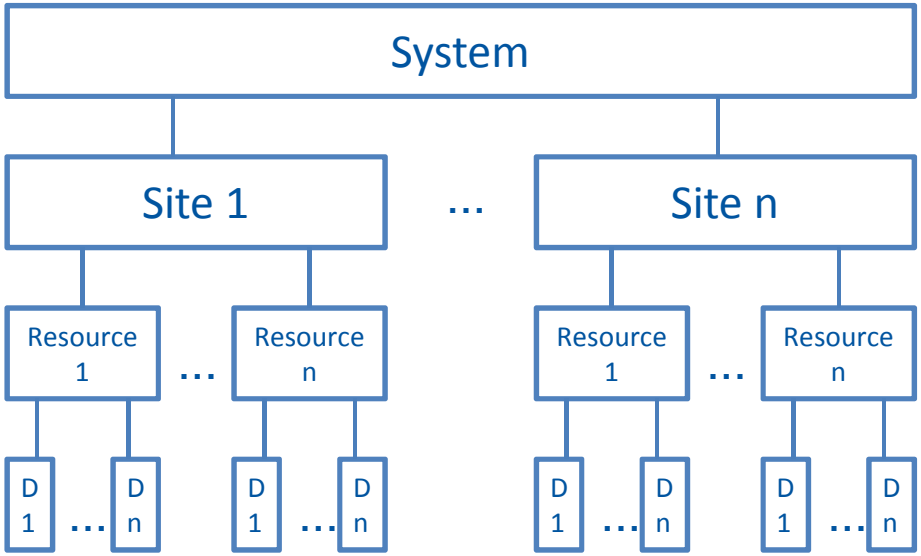
Steps of the adopted approach



Definition of system components and attributes



Class-design view of the system components. Each entity has its own attributes and inherits those of the parent.



General hierarchical view of the system components.

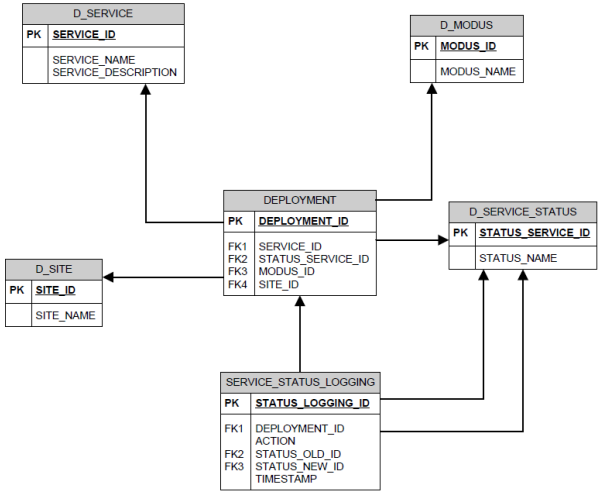
Metrics Ontology

Component/ Entity	Examples	Main attributes
Sites	CERN, BNL, TRIUMF, PIC, DESY	Site specific: reliability, availability; Inter-sites attributes: bandwith, latency
Resources	webserverXY@CERN, dbserverXY@BNL	Disk I/O, CPU, availability, load, usage statistics
Services	TAG database, extract, iELSSI	Network intensity factor, CPU intensity factor, disk I/O intensity factor, prioritzation factor
Deployments	iELSSI@webserverXY@CERN, TAG database@dbserverXY@BNL	Max. number of concurrent requests/sessions, availability, sucess/failure rate, performance, load, usage statistics

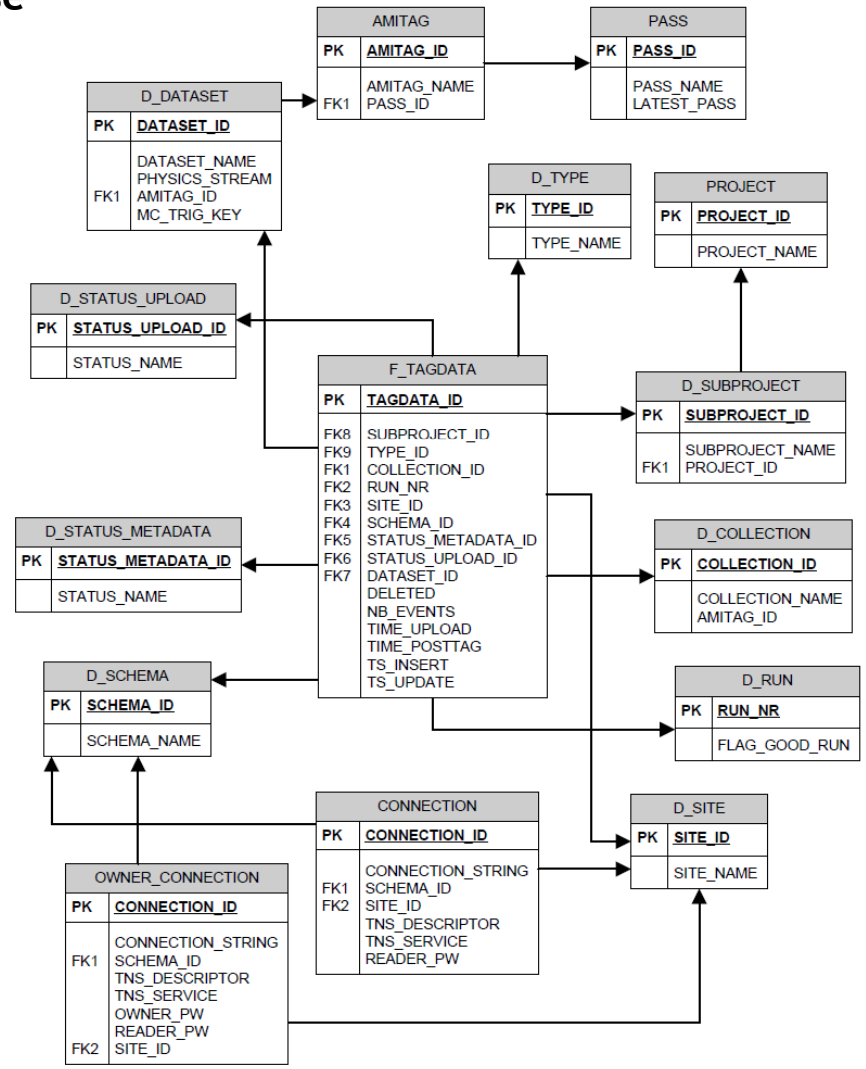
- All attributes mentioned above have a specific formal definition → details will be in the proceedings paper.

Logging of system activity

- **TASK = TAG Application Service Knowledgebase**
- Metadata repository recording:
  - The data distribution
  - All sites, resources, services, deployments and their metrics.
- Database schemas, datawarehouse-like star design
- Automatic updates via PL/SQL and Python API, through **active and passive monitoring.**



(Implementation of the entity „resource“ under construction)



Query profiles

- In order to build an appropriate optimization model, we have to make assumptions as regarding the system usage.
- TAG users: physicists who use the TAG system through several entry points and follow different usage workflows. Their behavior can be categorized into query profiles as follows:
  - **batch queries:** accessing TAG data in production jobs (mainly from event picking). These queries are done on a regular, daily basis.
  - **interactive queries:** individual users browsing through iELSSI and making interactive queries like counts, display results etc. These queries are sparse and less predictable, there are peaks before conferences and during tutorials.
  - **group production queries:** physics groups can run regular jobs to make bulk extractions based on specified cuts. This is less frequent than the batch query profile, and it might also have peaks before conferences.
- Each query profile is defined with characteristic access patterns and has different requirements. In the future, it should be possible to prioritize query profiles, e.g. depending on time periods.

Optimization objectives and algorithms

- From the system administration point of view, the value to maximize is the **throughput**, while performance considerations, such as the maximum time for processing (per data unit) are defined in **constraints**.
- In this context, throughput is defined as:

  *the average response time of **all** users over a time period.*

  *which is subject to be **minimzed**.*
- Approaches include:
  - Model a multidimension multichoice knapsack problem (take one item out of each knapsack)
  - Blackboard approaches
  - Game theory approaches

Simulation and Resource Planning

- Simulation framework, with variables:
  - Number of resources
  - Certain resource and deployment attributes
- Input: typical workload of a day, as recorded (e.g. number of calls per service with amount of data processed etc.)
- Processing: process the workload with the real setup and with changed parameters (e.g. additional resources, simulating higher CPU capacity etc.). Additionally, especially in the first phase, the optimization algorithm can be varied.
- Output: estimated response time per request, average, peaks etc.
- => The goal is to allow to make recommendations as regarding the resources, especially if the workload is expected to get higher and/or if a higher minimum number of concurrent requests is expected to be served within a reasonable, defined, time.

First and expected results

- Current status:
  - TASK implemented (data and services registry)
  - Some deployments already log continuously their activity into TASK, other to be included; off-line computation of aggregated metrics.
  - Formal model under definition
- Implemented so far:
  - automated database selection from iELSSI, based mainly on network latency parameters and the browser location the user is connected to – other attributes to be included step by step.
  - [comparative db activity graphs to be included here!]
- Next steps:
  - Complete deployments logging
  - Simulation setup
- Expected results:
  - conforming to the goals stated on slide 6, with formal justification.
  - Framework for making resource optimization recommendations.

Conclusions

- Quite some assumptions have to be made to build the model, compute attributes and define query profiles.
  - We are thus mostly not speaking about absolute values, but approximations!
- In a long-lasting experiment like ATLAS, query profiles and optimization objectives are subject to changes, the service selection model should thus be open to such evolutions. The proposed approach is dynamic as regarding:
  - the query profiles and their prioritization
  - the prioritzation of services – this is an attribute of the entity *service* and can be changed
  - the underlying infrastructure – sites and resources can appear and disappear on the fly, with the restriction that they need to be registered and set up to log their activity.
- The special view-point of the presented approach lies in the fact that the system is subject to centralized control, which offers possibilities that traditional distributed systems with federated control do not offer.