



UNIVERSITY OF DORTMUND

ROBOTICS RESEARCH INSTITUTE
INFORMATION TECHNOLOGY SECTION



HEP CG Scheduling Architecture

Lars Schley
Uni Dortmund, IRF

15/03/2007



Goals



Current situation:

- Allocation of jobs to sites may not incorporate knowledge about data availability
 - Files may be on tertiary storage and first access is delayed
 - Storage element does not provide advanced planning features
-
- More precise planning and prediction of file availability prior to job allocation to compute elements
 - Better integration of local job and data scheduling to improve response times and through-put



- Co-scheduling of jobs and data
- Job types that can be distinguished
 - *CPU intensive algorithms use small input data sets to produce large sets of simulation data:*
 - Due to the high computational resource consumption the resulting data sets are valuable
 - Resulting data sets have to be distributed → coherence?
 - *Reconstruction / Reprocessing:*
 - Access to the simulation data is needed
 - Use of CPU intensive algorithms
 - *Analyze jobs:*
 - Quantity and local occurrence are not predictable
 - Use of data sets from experiments and/or simulations
 - Variable in CPU utilization and resulting data sets



HEP Development



- Our work will be based on the developments in the LCG environment
 - Initially, LCG 2.X software
 - As soon as sensible, switch over to web service based gLite (target middleware)
- Extension of the approved dCache system
- Usage of Globus components only if/where necessary

- **Desirable:** cooperation, discussion and maybe integration of our (future) research and corresponding results into the LCG community
 - to assure maintenance



Extended Scheduling System



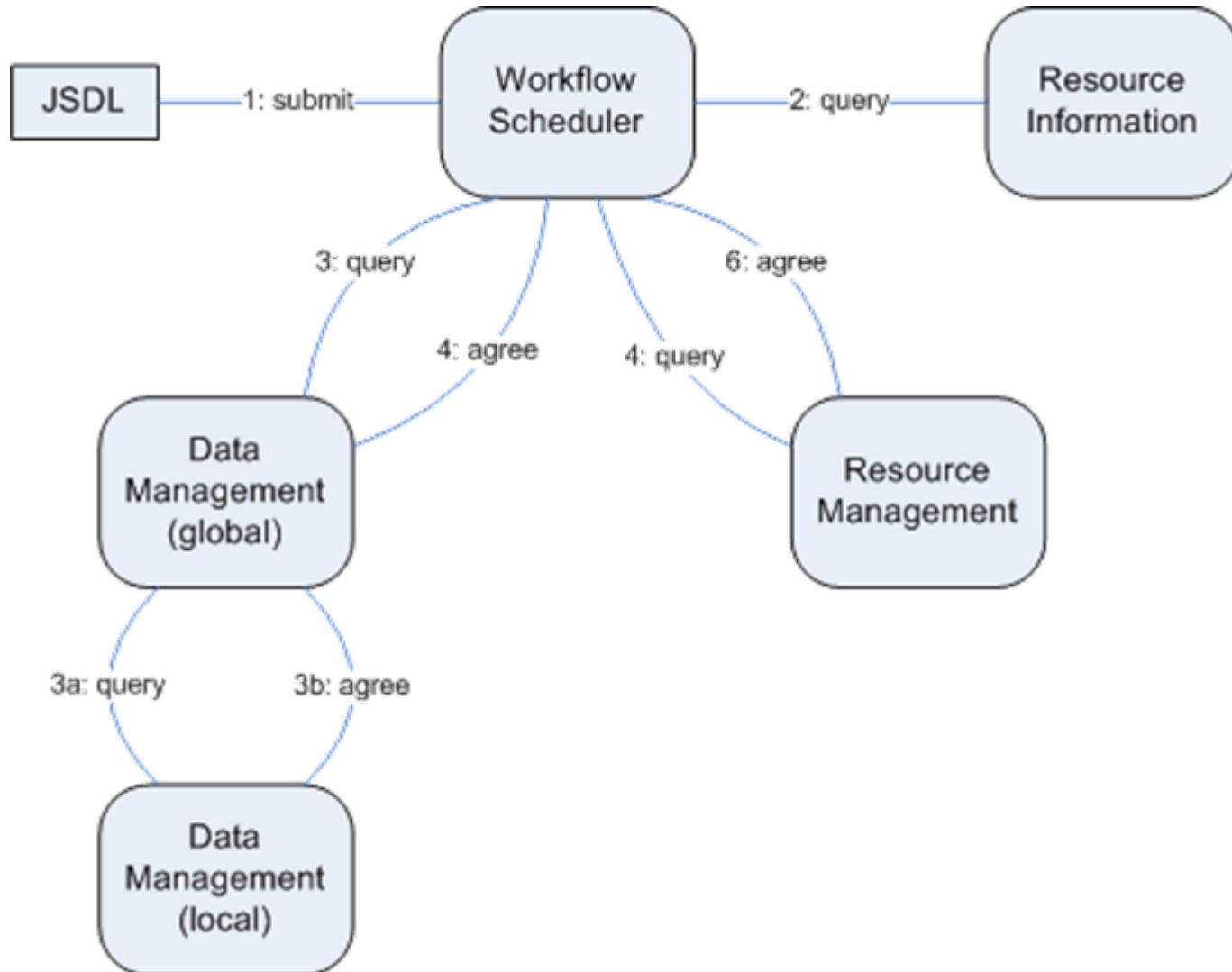
- Receives jobs from the user (JDL)
- Interacts with the Monitoring and Discovery Service (MDS) to find fitting resources
 - dependent on the submitted JDL
- Interacts with a File Catalog (LFC) to translate LFNs, PFNs and GUIDs
- System makes use of the local schedulers in the extended CEs and SEs
 - Add SE interface to communicate about future data/storage requests
 - Add CE interface to communicate about future job executions
- If replication has to be accomplished a corresponding replication scheduler will be invoked.
- **Desirable:** Possible extension/addition to current WMS development?



- Projects have different objectives and requirements
 - **C3Grid:** Data preprocessing,
WS/GT4 Infrastructure
 - **HEP:** Connection of the SE to optimize data access on HSM
LCG / gLite Infrastructure
- But, both projects focus on co-scheduling of jobs and data
- Despite those differences we can develop and use common definitions, models and strategies

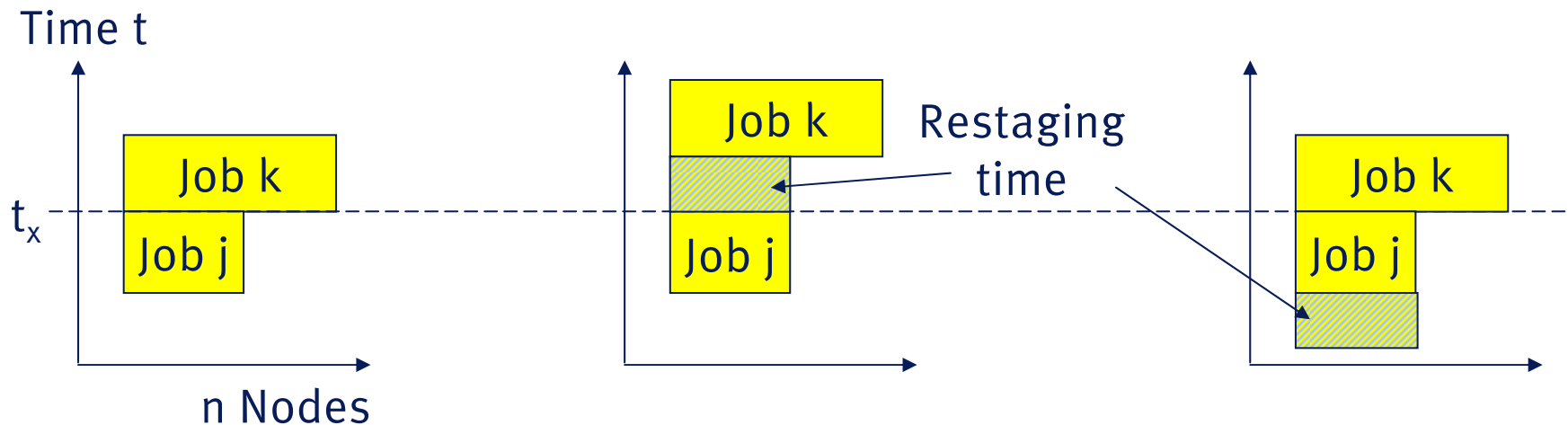


Example: Flow of Interaction in C3Grid





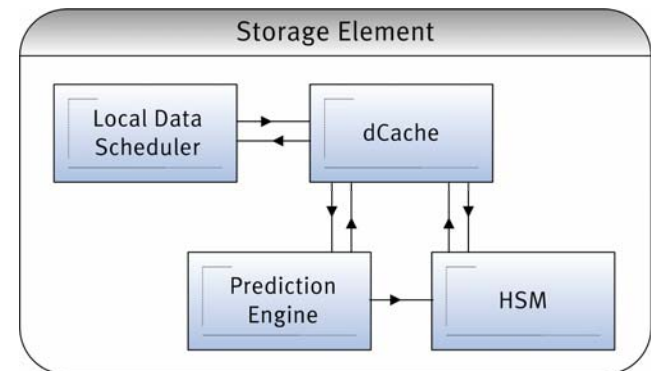
- Job j tries to access the required data sets
 - sets are not “really” present
 - sets are located on tertiary storage



- SE should be included into scheduling process to avoid idle times
 - Also SE has to be schedulable

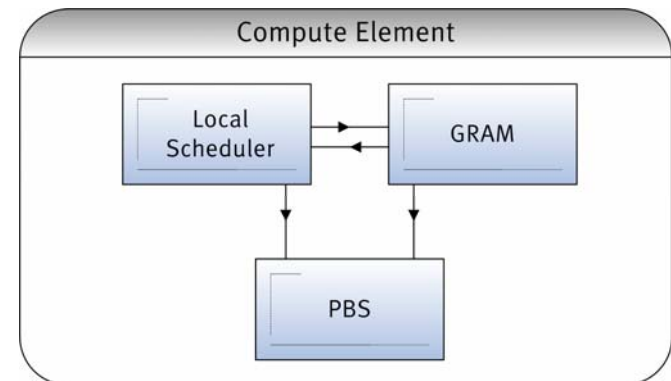


- Due to the use of HSM, files appear “available” on a local file system → essentially located on tertiary storage
 - Restaging on demand can take a long period of time
 - ready to run jobs have to stay idle until the recovery of prerequisite data
 - Restaging time is difficult to predict
- Ext. of the SE (dCache & HSM) by a prediction engine
 - Estimation of restoration times: When will a file be accessible?
- Ext. of the SE by a local data scheduler (LDS) to a schedulable resource
 - LDS has to deal with space reservation (dCache support)
 - LDS communicates with dCache: When is a file accessible?
 - dCache and LDS need agreement mechanisms





- Extension of the CE by a local scheduler to a schedulable resource
- Functions of the local scheduler:
 - Query of the dynamic state: When can the CE execute the next job?
 - Agreement mechanism for the execution of jobs
 - Interaction with SE: Is the required data finally available on the SE?
- Architecture is based on the current CE
- Torque controlled by GRAM

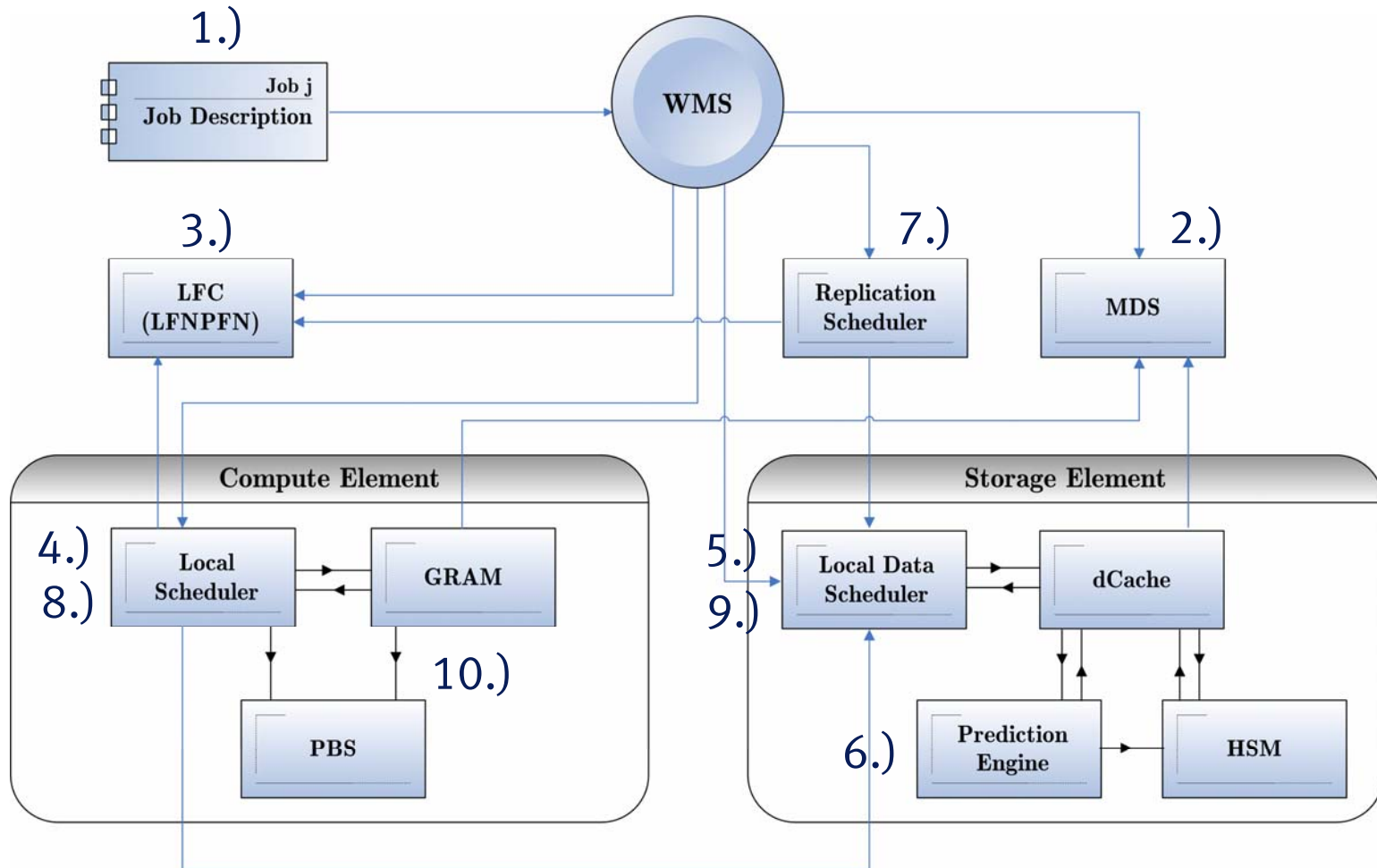




Central Scheduling System



- Receives jobs from the user (JDL)
- Interacts with the Monitoring and Discovery Service (MDS) to find resources
 - dependent on the submitted JDL
- Interacts with a File Catalog (LFC) to translate LFNs, PFNs and GUIDs
 - Where are the files located?
- System makes use of the local schedulers in the extended CEs and SEs
 - Which is the earliest start time for a job?
 - Has file replication to take place?
 - Are the files during execution time immediately accessible?
- If replication has to be accomplished a replication scheduler will be invoked





- Both projects focus on co-scheduling of jobs and data

Job Scheduling Model

Fundamentals of Scheduling Strategies

Usage of interfaces,
which hide the
„real“
infrastructure

pluggable

Community specific
provider implement
those generic
interfaces

Scheduling Algorithm

Scheduling Algorithm

Implementation

Implementation

Integration

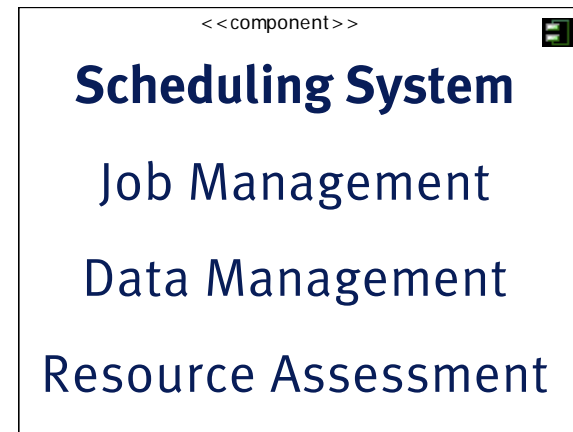
Integration

C3Grid CG

HEP CG

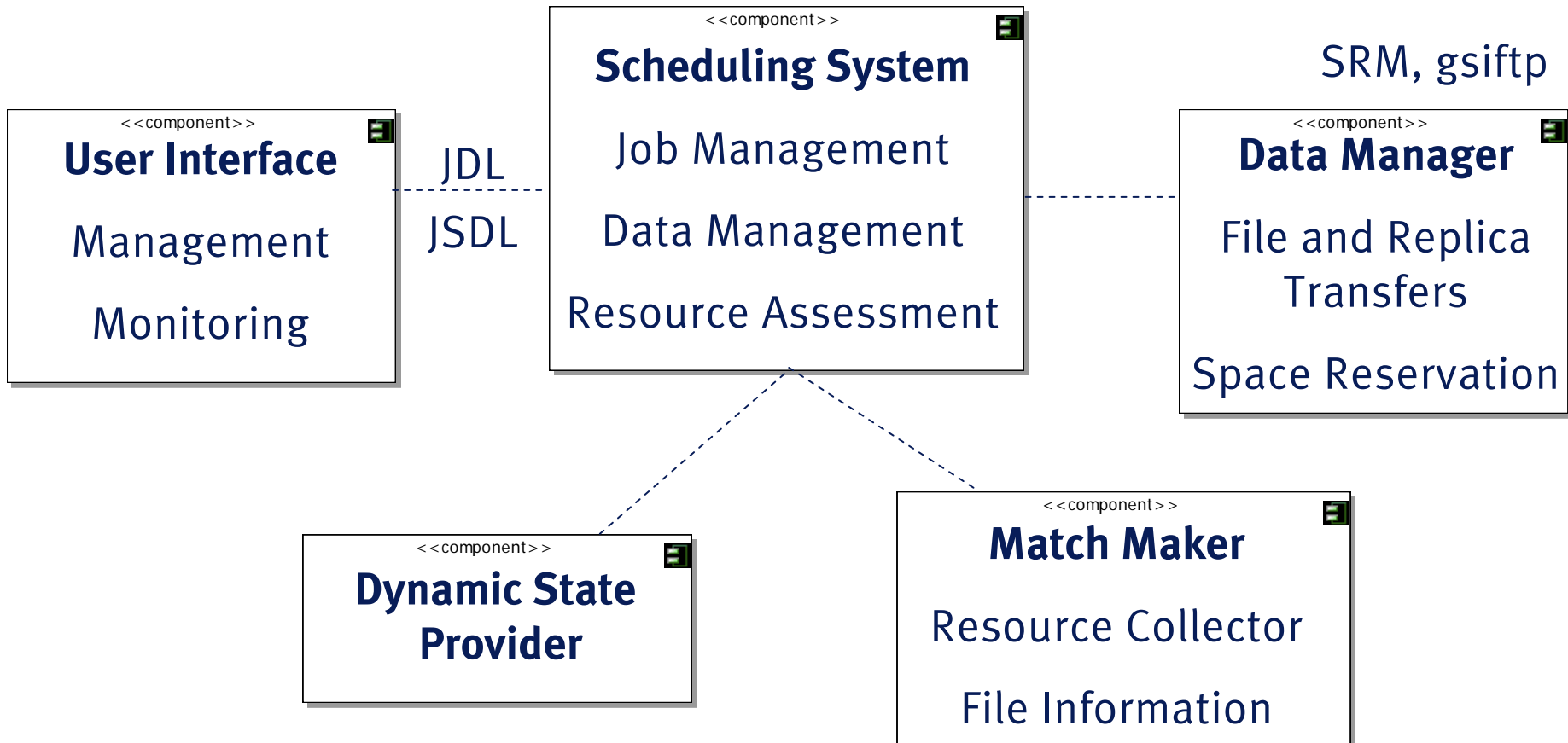


- System will be independent from special infrastructures
 - Abstract interfaces for resources, information system and file catalog
 - Definition of an abstract resource object encapsulates the HEP CE
- Job Management:
 - Job submission
 - Job control (e.g. and cancellation)
 - State Monitoring
- Data Management:
 - File Transfer Service
 - Replication Management
 - Space Reservation
- Resource Assessment:
 - Dynamic state information from SEs and CEs



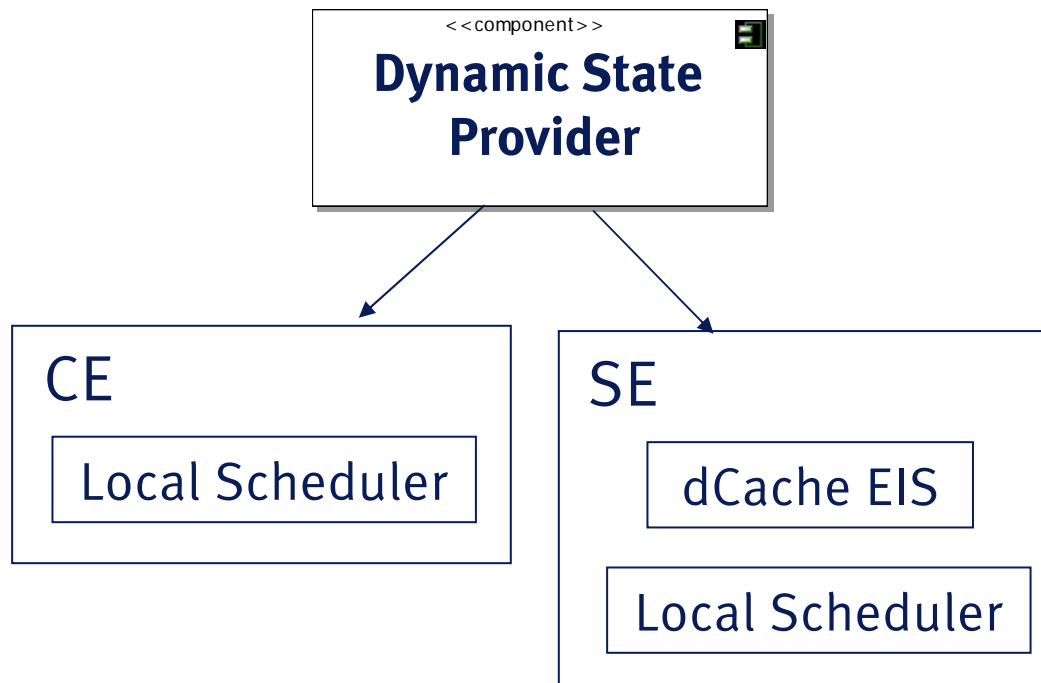


Scheduler internal framework HEP CG & C3Grid



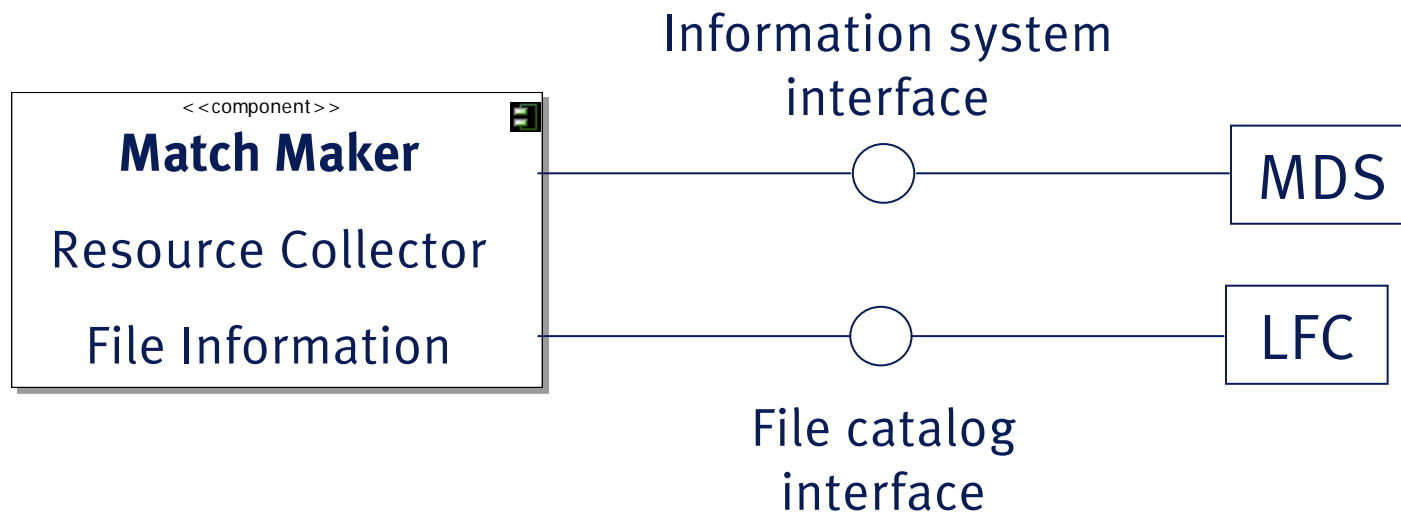


- Local Schedulers of SE and CE can be interrogated by the scheduling system through the dynamic state provider
- Usage of dCache Extended Information Service (EIS)





- Match Maker can query MDS and LFC through interfaces
- Other possible providers:
 - ➔ C3 information system, MDS4
 - ➔ Simulation provider for benchmarks





Questions?

