CPPC-G: Fault-Tolerant Parallel Applications on the Grid

Daniel Díaz, Xoan Pardo, María J. Martín, Patricia González, Gabriel Rodríguez, Juan Touriño & Ramón Doallo

Grupo de Arquitectura de Computadores

Universidade da Coruña

Contents

1. Fault-tolerance in Grid environments
2. CPPC
3. CPPC-G
   - Overview
   - Services
4. Conclusions
5. Future work

Fault-tolerance in Grid environments (1)

- Fault-tolerance is a desirable feature for applications
  - It becomes vital for long applications whose time to complete is close to the Mean Time to Failure (MTTF) of the underlying hardware
  - One should take into account the possibility of problems when submitting jobs to remote computing resources…

Fault-tolerance in Grid environments (2)

- One way to achieve fault-tolerance is to perform checkpointing
  - To periodically save the application state in stable storage…
  - …so that interrupted executions can be restarted from the last saved state

Fault-tolerance in Grid environments (3)

- Checkpointing across the Grid
  - The generated checkpoint files must be **portable**
    - Able to be used with different architectures and/or operating systems
  - Some mechanism must exist for replicating the checkpoint files in a safe backup place as they are created
    - …and for moving them to a new machine in the case of a restart

CPPC (1)

- What is CPPC?
  - A checkpointing library for message-passing parallel applications
  - A compiler that helps the programmer to automate the development of applications that use it

CPPC (2)

- CPPC features
  - Distributed checkpointing, with compile-time coordination
    - The programmer must indicate “safe points” in the application code at which to take checkpoints
    - All processes in a parallel application take the checkpoints at the same relative locations in the code
    - After a failed execution, a consistent set of checkpoint files is selected, and the execution can be restarted from there

CPPC (3)

- CPPC features (cont.)
  - Only portable application state is saved. Non-portable state is recovered through re-execution
  - Checkpointing performed at the variable level
    - The hardware context of the machine is not saved
  - The checkpoint files are written in a portable format
    - Portability is achieved by using the HDF5 library for storage of scientific data

CPPC-G overview (1)

- What is CPPC-G?
  - A set of Grid Services that handle execution of CPPC applications over the Grid
    - Implemented with the Globus Toolkit 4
    - It’s built on top of existing services (WS-GRAM, RFT, MDS) whenever possible

CPPC-G overview (2)

- Why CPPC-G?
  - Existing services allow for remote execution but more functionality is needed
    - Handling of checkpoints
    - Restart capability
    - Automatic resource selection
      - Metaschedulers handle this (at the moment not part of Globus)

CPPC-G overview (3)

- CPPC-G features
  - Automatic selection of Grid computing resources
    - According to a scheduling need supplied by the user
  - Remote execution of CPPC applications
    - Each application is submitted to only one computing resource
    - Parallel processes are not distributed across the Grid, instead they are spawned internally on the computing resource (e.g. MPI applications in a cluster)

CPPC-G overview (4)

- CPPC-G features (cont.)
  - Monitoring of running CPPC applications and detection of failed executions
    - Done by periodically polling the computing resource
  - Periodic replication of generated checkpoint files
    - On a safe backup location
    - Checkpoint metadata is collected

CPPC-G overview (5)

- CPPC-G features (cont.)
  - Restart of failed applications
    - A consistent set of checkpoint files is selected based on the collected metadata
  - Cleanup of checkpoint files
    - Checkpoint files are automatically deleted after they become obsolete
      - When a new consistent set is composed
      - When the application finishes succesfully

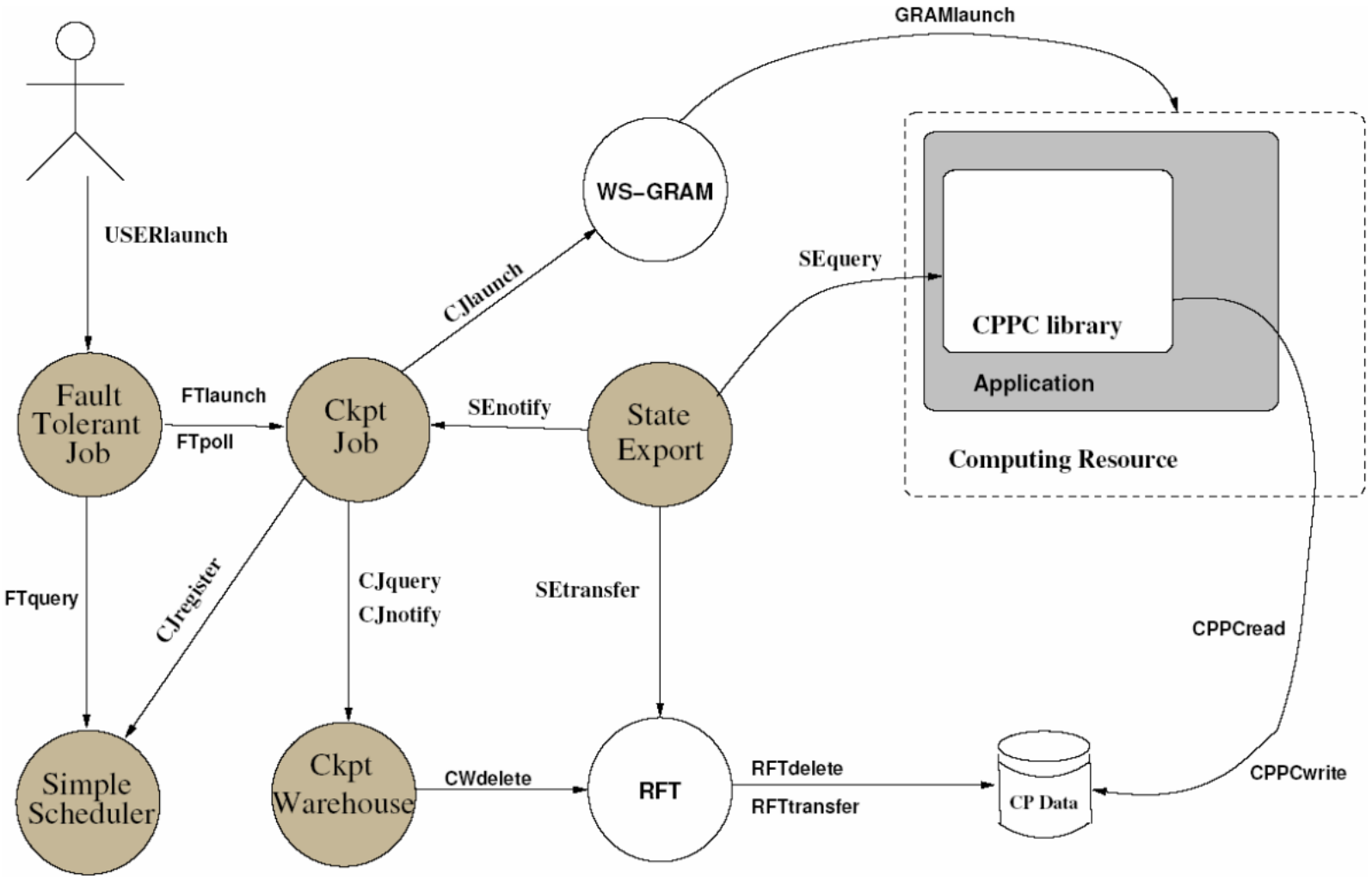
CPPC-G overview (6)

- CPPC-G design principles
  - A set of Grid services instead of one or two monolithic services
    - Allows the user to employ only a subset of the system functionality
    - Allows more flexibility to the administrator when deploying the services
    - Trade-off between increased flexibility and the overhead of each new service

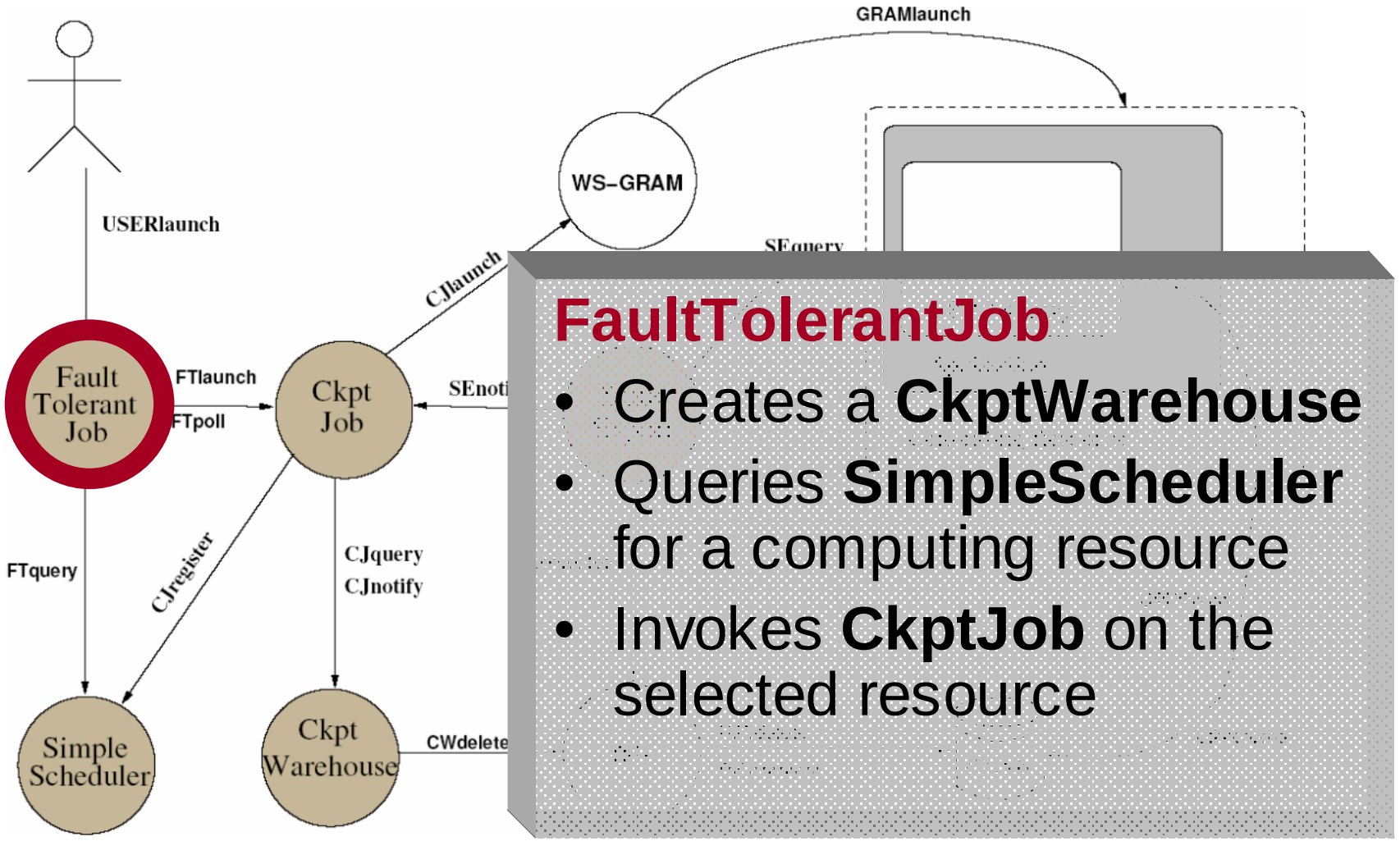
CPPC-G overview (7)

- CPPC-G design principles (cont.)
  - User has maximum control over his delegated credentials
    - Services do not re-delegate user credentials unless absolutely neccesary
      - When the computing resource is not known beforehand
    - More secure, but more cumbersome for the user

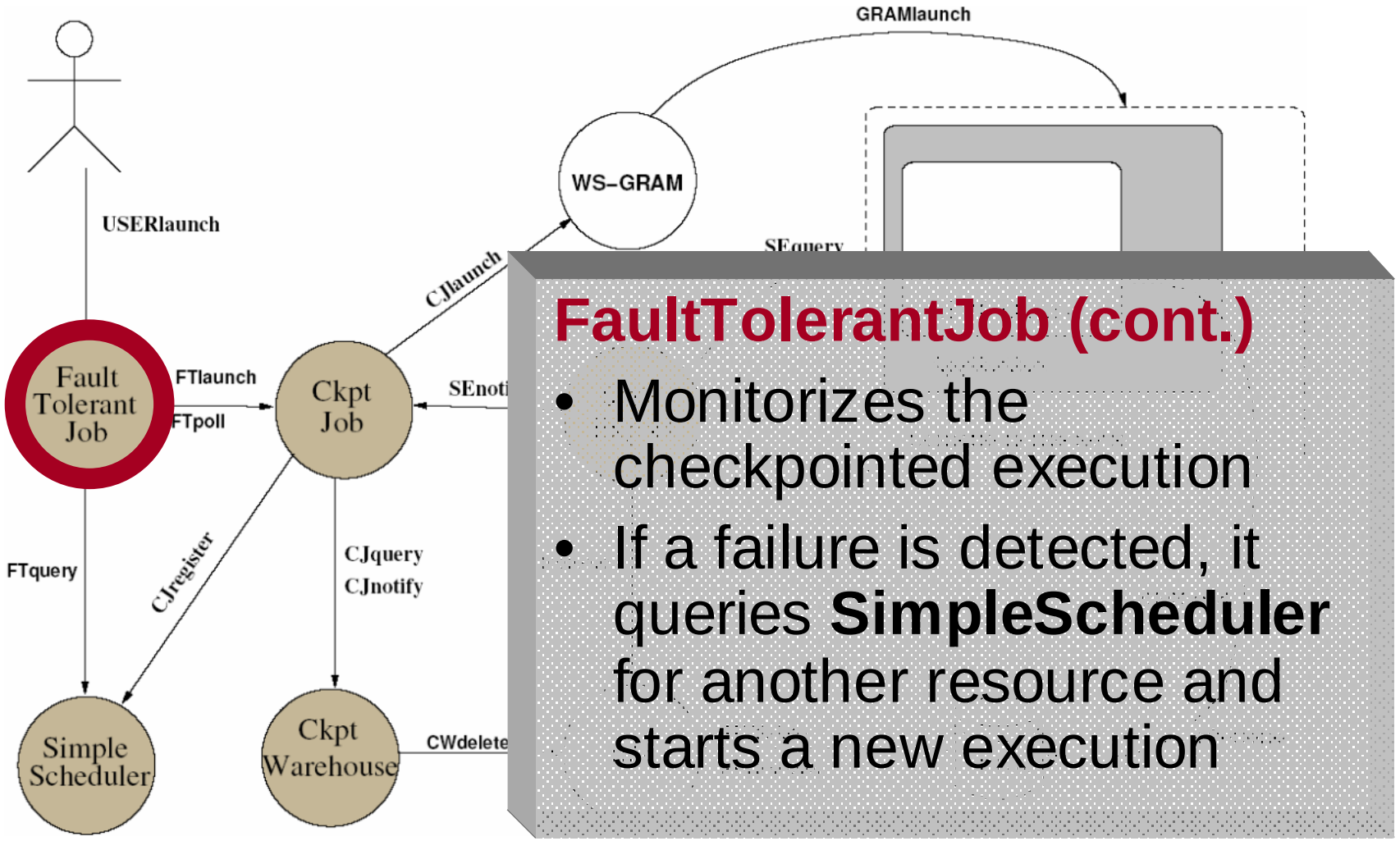
CPPC-G services (1)



CPPC-G services (2)



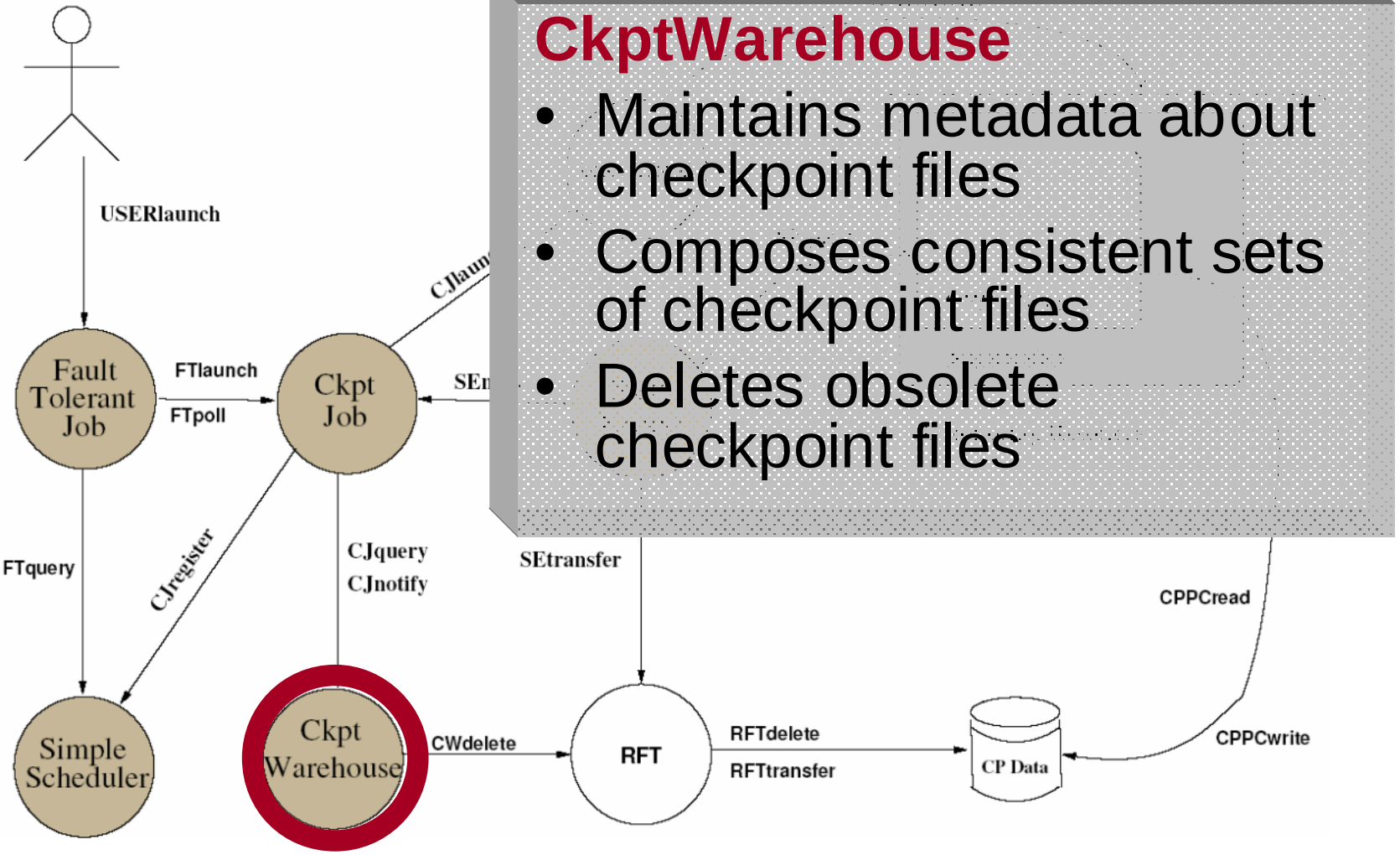
CPPC-G services (3)



FaultTolerantJob (cont.)

- Monitorizes the checkpointed execution
- If a failure is detected, it queries **SimpleScheduler** for another resource and starts a new execution

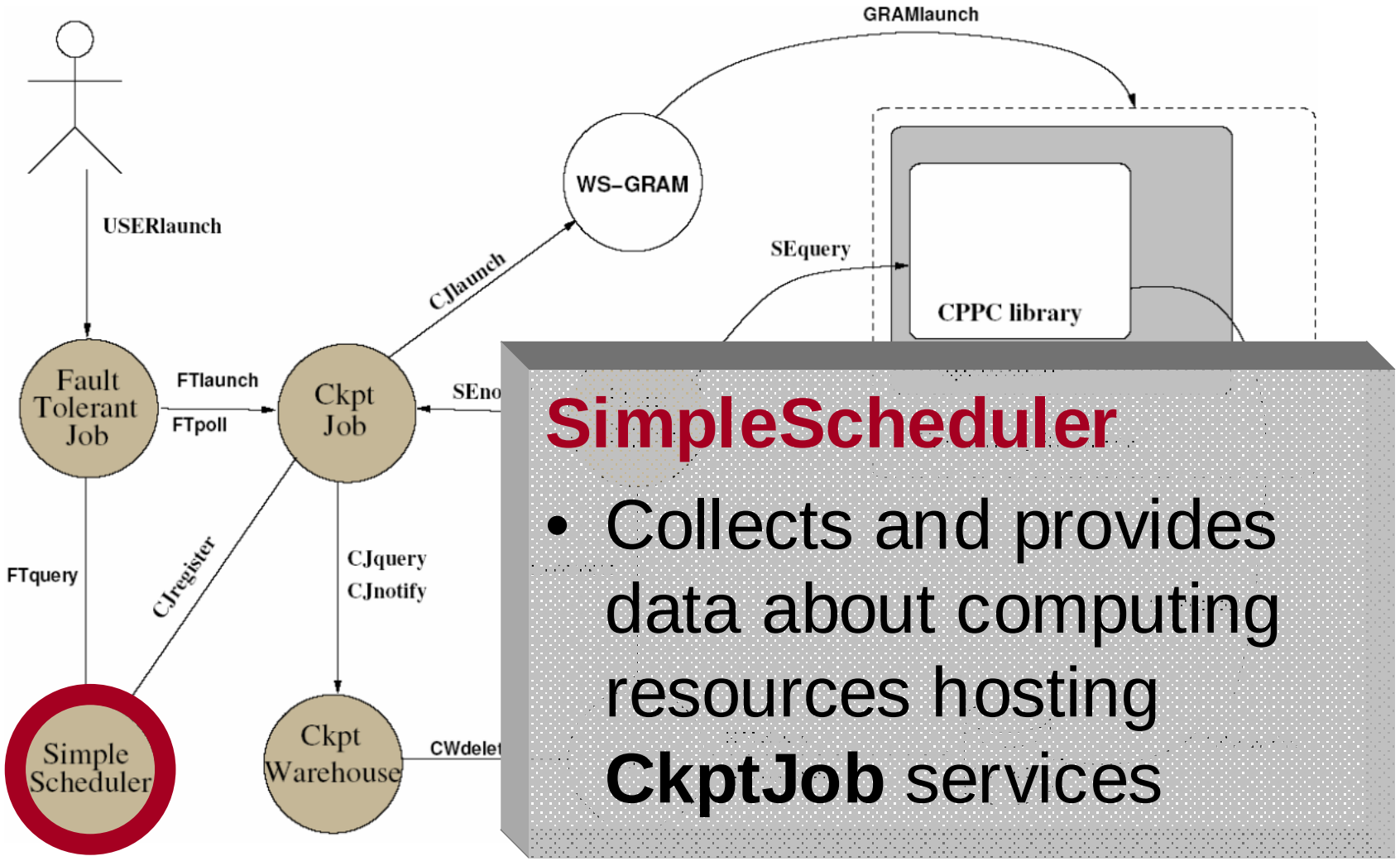
CPPC-G services (4)



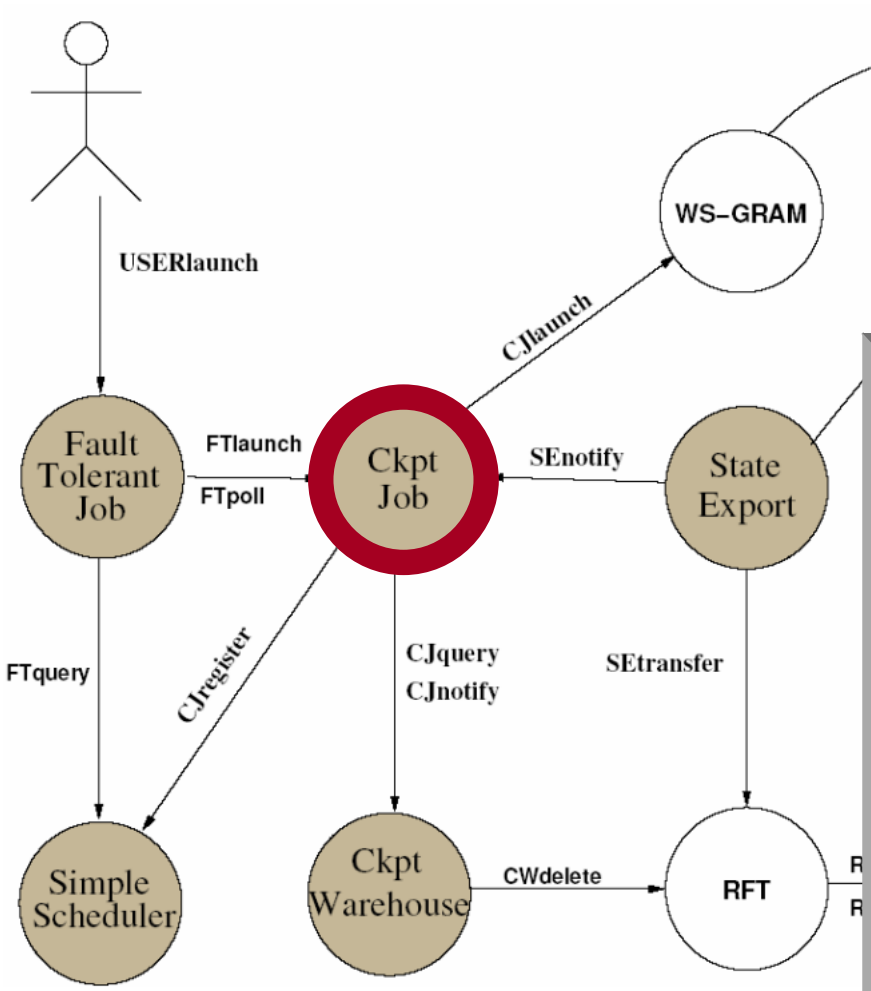
CkptWarehouse

- Maintains metadata about checkpoint files
- Composes consistent sets of checkpoint files
- Deletes obsolete checkpoint files

CPPC-G services (5)



CPPC-G services (6)



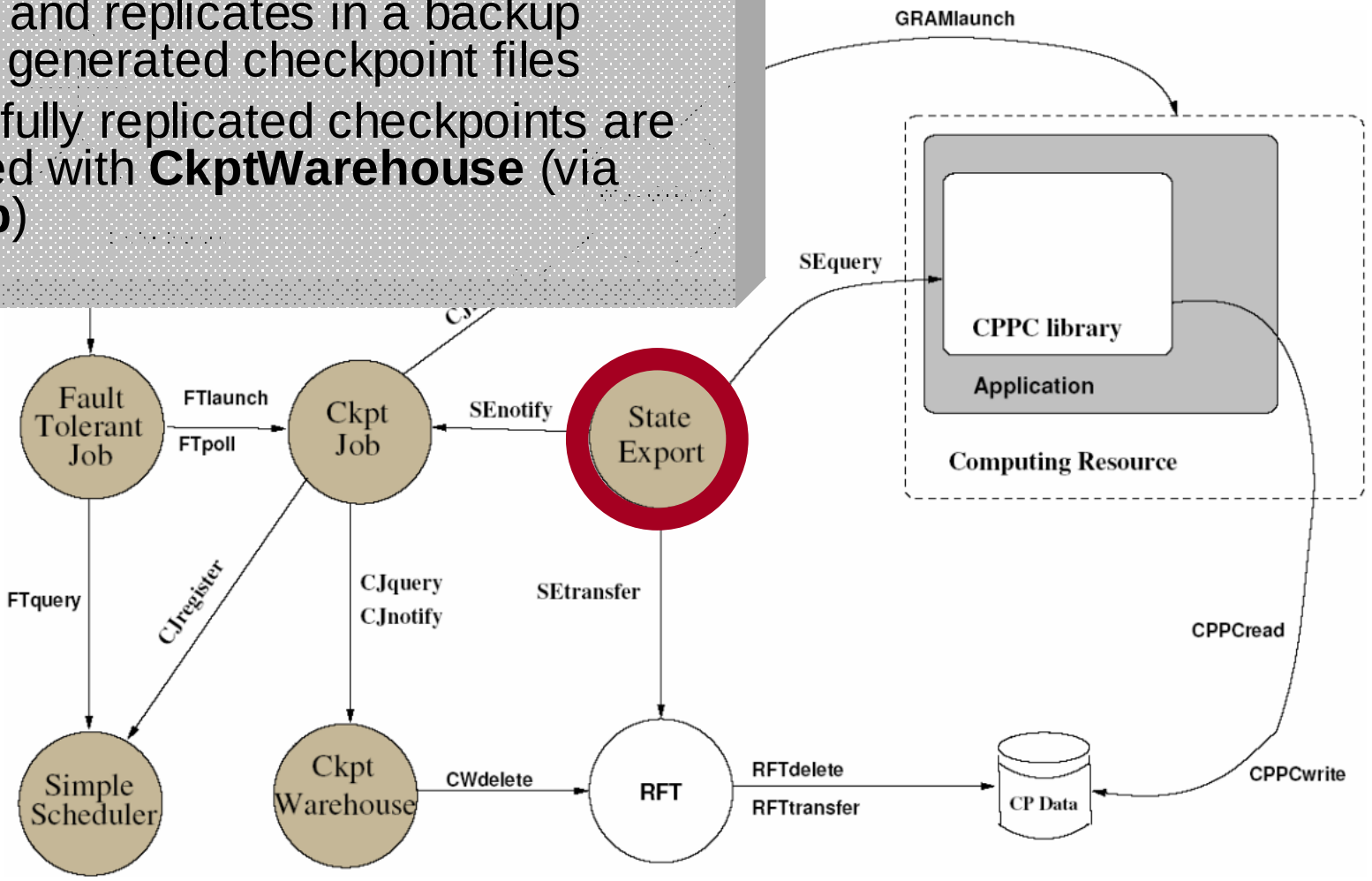
CkptJob

- Manages the checkpointed execution of **CPPC** applications
  - On activation, it registers itself with the **SimpleScheduler**
  - When invoked, it gets the last consistent set of checkpoint files from **CkptWarehouse**
  - Submits the **CPPC** application to the local job manager

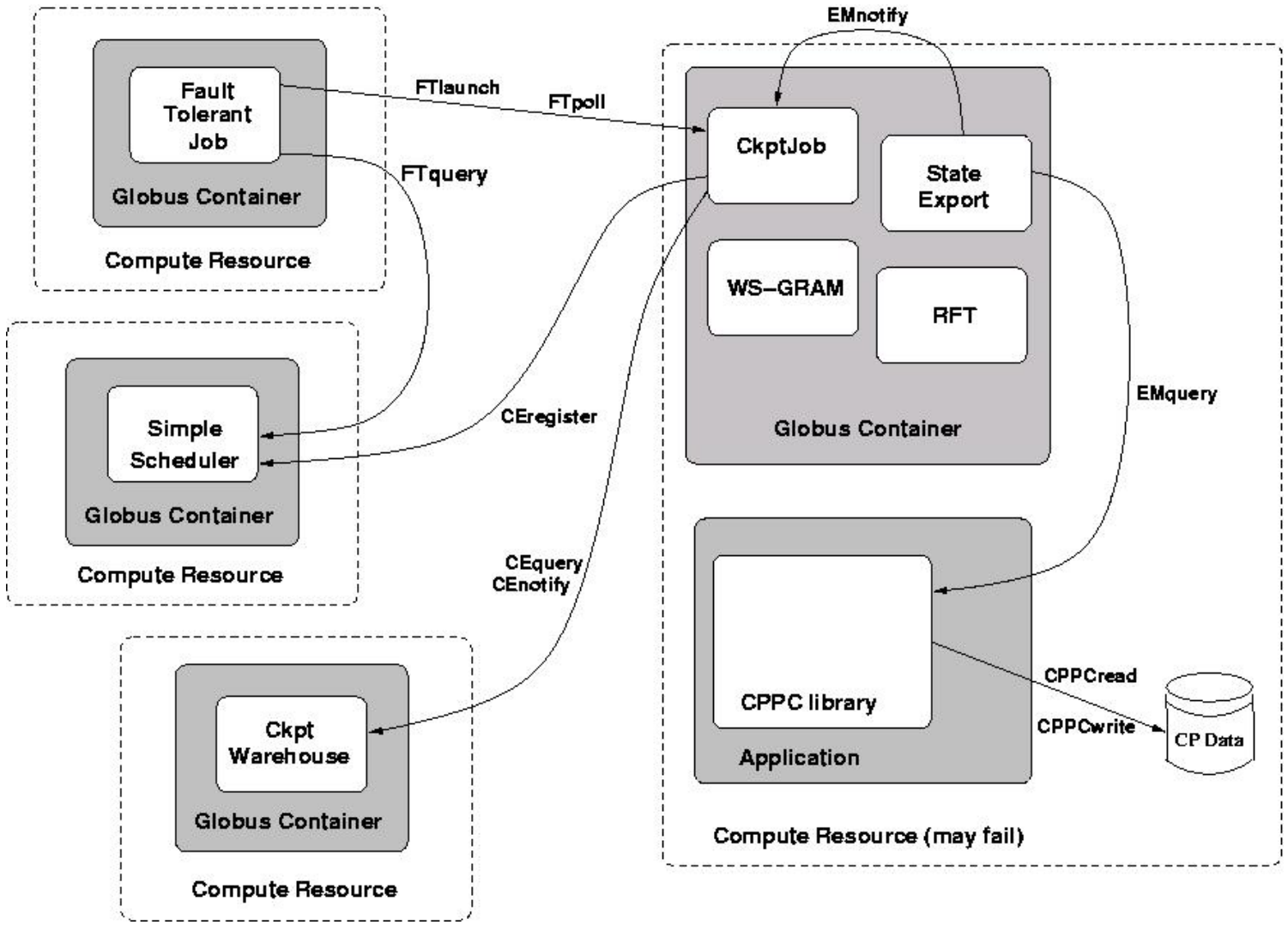
CPPC-G services (9)

StateExport

- Detects, and replicates in a backup location, generated checkpoint files
- Successfully replicated checkpoints are registered with **CkptWarehouse** (via **CkptJob**)



CPPC-G, service by service (12)



Conclusions

- **CPPC-G** is a set of Grid services that supports the fault-tolerant execution of **CPPC** applications across the Grid
  - Finds avaliable computing resources according to the user's needs
  - Launches and monitorizes the checkpointed execution of **CPPC** applications
  - Restarts them in case of failure (possibly in a different computing resource)
  - Modular, flexible and secure architecture

Future Work

- Work in progress, near completion!
- Make the architecture fault-tolerance itself
- More reliable error detection techniques
- Automatic selection of the checkpoint backup site
- Integration with existing metaschedulers

CPPC-G:
Fault-Tolerant Parallel Applications on the Grid

Daniel Díaz

ddiaz@udc.es

Grupo de Arquitectura de Computadores

Universidade da Coruña