

User-driven resource selection in GRID superscalar

Last developments and future plans
in the framework of CoreGRID

Rosa M. Badia

Grid and Clusters Manager

Barcelona Supercomputing Center

<http://www.coregrid.net>

rosa.m.badia@bsc.es

Outline

1. GRID superscalar overview
2. User defined cost and constraints interface
3. Deployment of GRID superscalar applications
4. Run-time resource selection
5. Plans for CoreGRID

1. GRID superscalar overview

Programming environment for the Grid

Goals:

- Grid as transparent as possible to the programmer

Approach

- Sequential programming (small changes from original code)
- Specification of the Grid tasks
- Automatic code generation to build Grid applications
- Underlying run-time (resource, file, job management)

1. GRID superscalar overview: interface

```
GS_On();  
for (int i = 0; i < MAXITER; i++) {  
    newBWd = GenerateRandom();  
    subst (referenceCFG, newBWd, newCFG);  
    dimemas (newCFG, traceFile, DimemasOUT);  
    post (newBWd, DimemasOUT, FinalOUT);  
    if (i % 3 == 0) Display(FinalOUT);  
}  
  
fd = GS_Open(FinalOUT, R);  
printf("Results file:\n"); present (fd);  
  
GS_Close(fd);  
  
GS_Off();
```

1. GRID superscalar overview: interface

```
void dimemas(in File newCFG, in File traceFile, out File DimemasOUT)
{
    char command[500];

    putenv("DIMEMAS_HOME=/usr/local/cepba-tools");
    sprintf(command, "/usr/local/cepba-tools/bin/Dimemas -o %s %s",
            DimemasOUT, newCFG);

    GS_System(command);
}
```

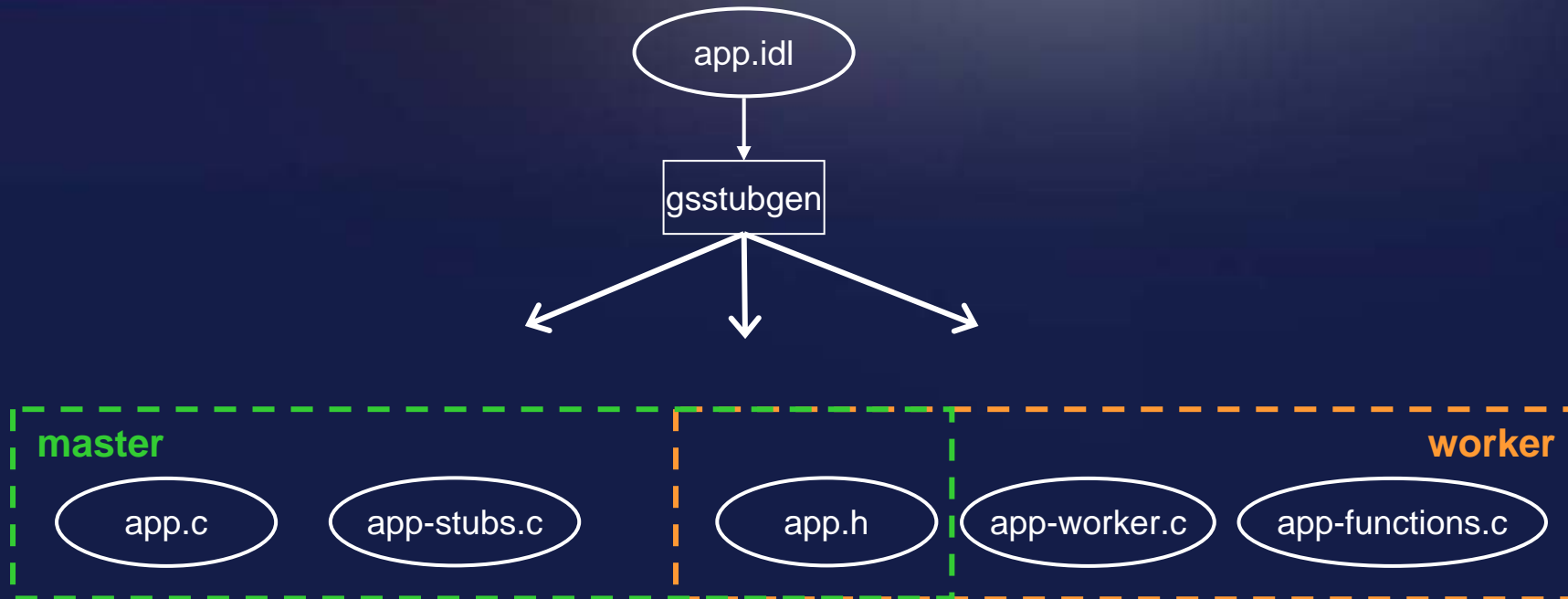
```
void display(in File toplot)
{
    char command[500];

    sprintf(command, "./display.sh %s", toplot);
    GS_System(command);
}
```

1. GRID superscalar overview: interface

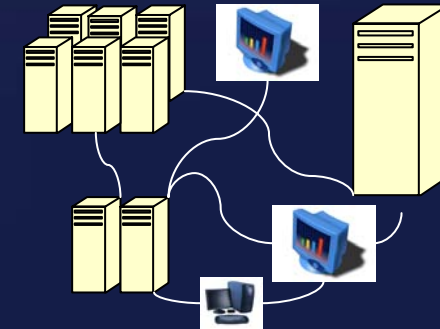
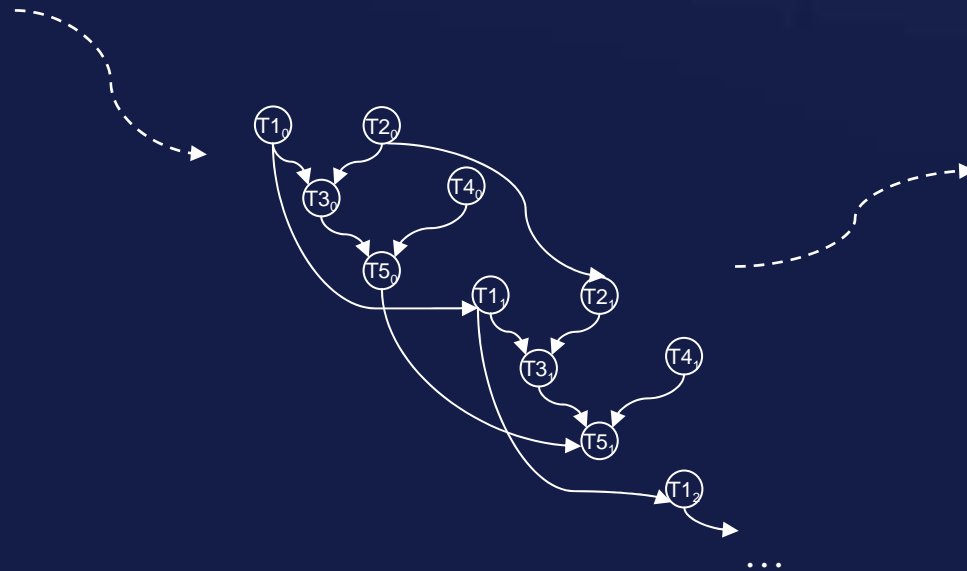
```
interface MC {  
    void subst (in File referenceCFG, in double newBW, out File newCFG);  
    void dimemas (in File newCFG, in File traceFile, out File DimemasOUT);  
    void post (in File newCFG, in File DimemasOUT, inout File FinalOUT);  
    void display (in File toplot)  
};
```

1. GRID superscalar overview: code generation



1. GRID superscalar overview: behaviour

```
for (int i = 0; i < MAXITER; i++) {
    newBwd = GenerateRandom();
    substitute ("nsend.cfg", newBwd, "tmp.cfg");
    dimemas ("tmp.cfg", "trace.trf", "output.txt");
    postprocess (newBwd, "output.txt", "final.txt");
    if(i % 3 == 0) display("final.txt");
}
```



1. GRID superscalar overview: runtime features

Data dependence analysis

File renaming

Shared disks management

File locality exploitation

Resource brokering

Task scheduling

Task submission

Checkpointing at task level

Exception handling

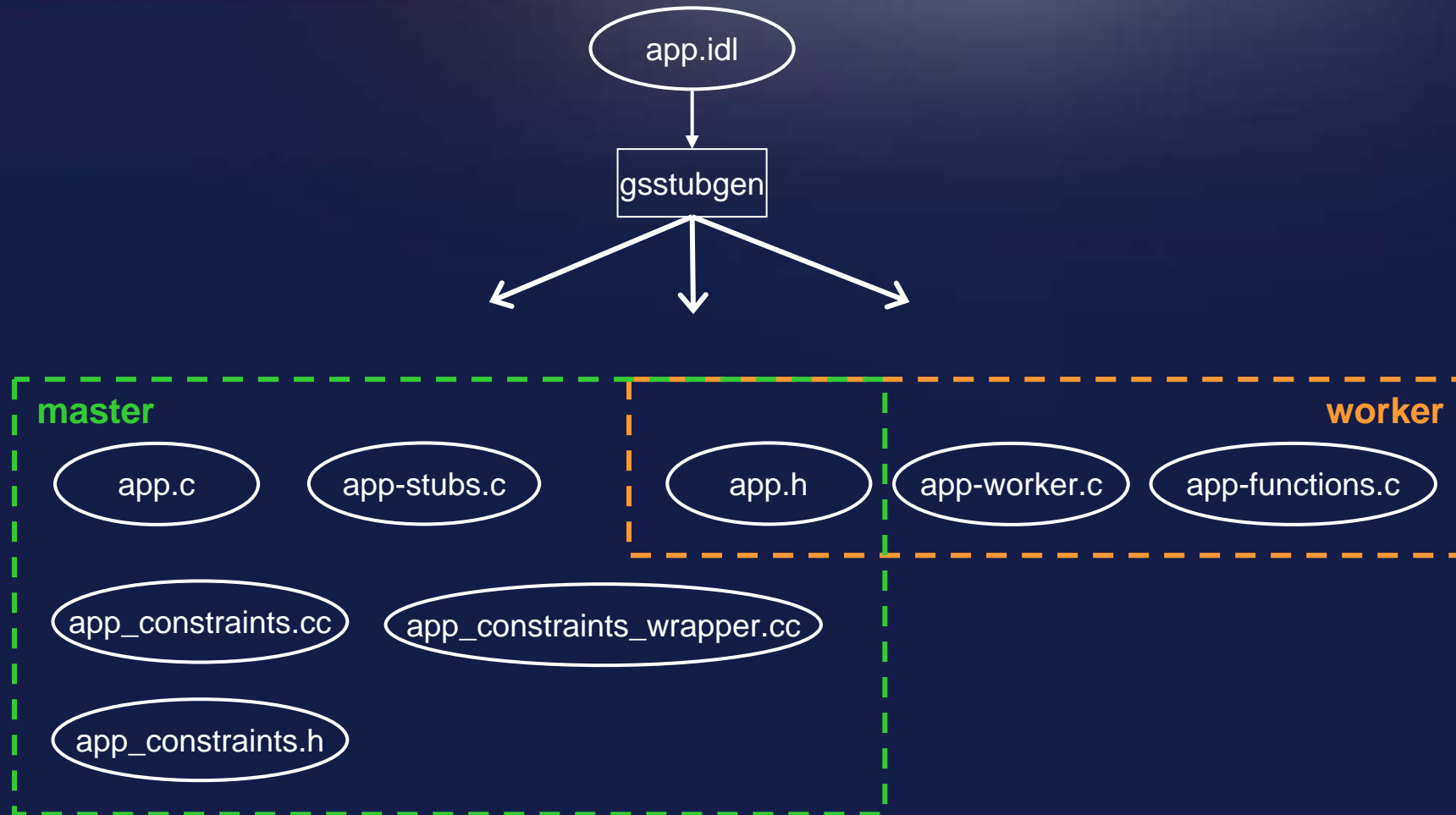
Current version over Globus 2.x, using the API

File transfer, security, ... provided by Globus

Ongoing developments of versions:

- Ninf-g2
- ssh/scp
- GT4

2. User defined cost and constraints interface



2. User defined cost and constraints interface

File `app_constraints.cc` contains the interface of functions for

- Resource constraints specification
- Performance cost estimation

Sample default functions:

```
string Subst_constraints(file referenceCFG, double
    seed, file newCFG) {
    string constraints = "";
    return constraints;
}
double Subst_cost(file referenceCFG, double seed,
    file newCFG) {
    return 1.0;
}
```

2. User defined cost and constraints interface

Users can edit and specify constraints and performance cost for each function

- Constraints syntax: Condor ClassAds
- Performance cost syntax: pure C/C++

```
string Dimem_constraints(file cfgFile, file traceFile)
{
    return "(member(\"Dimemas\", other.SoftNameList) &&
           other.OpSys == \"Linux\" && other.Mem > 1024 )"
}
double Dimem_cost(file cfgFile, file traceFile)
{
    double complexity, time;

    complexity = 10.0 * num_processes (traceFile) +
    3.0 * no_p_to_p (traceFile) + no_collectives (traceFile) +
    no_machines(cfgFile);
    time = complexity / GS_GFlops();
    return(time);
}
```

3. Deployment of GRID superscalar applications

Java based GUI

Allows GRID resources specification: host details, libraries location...

Selection of Grid configuration

Grid configuration checking process:

- Aliveness of host (ping)
- Globus service is checked by submitting a simple test
- Sends a remote job that copies the code needed in the worker, and compiles it

Automatic deployment

- Sends and compiles code in the remote workers and the master

Configuration file generation

Deployment Center

File View Help

Certificate validity: 0d 21:32

Global host configuration

| Host | Min. port | Max. port |
|----------------------|-----------|-----------|
| kadesh.cepba.upc.es | 0 | 65535 |
| kandake.cepba.upc... | 0 | 65535 |
| khafre.cepba.upc.es | 20340 | |
| kharga.cepba.upc.es | 0 | |
| pcboada.ac.upc.edu | 0 | |

Modify worker

Compilation environment parameters

General options Queues Software packages

| Available | Software package |
|-------------------------------------|------------------|
| <input type="checkbox"/> | Dimemas |
| <input checked="" type="checkbox"/> | GAMESS |
| <input checked="" type="checkbox"/> | Paramedir |

Time

| | |
|------------------------------|--|
| Tue Mar 08 10:04:53 CET 2005 | Startup check start |
| Tue Mar 08 10:04:54 CET 2005 | Startup check finis |
| Tue Mar 08 10:04:54 CET 2005 | Started checking h |
| Tue Mar 08 10:04:54 CET 2005 | Started checking h |
| Tue Mar 08 10:04:54 CET 2005 | Started checking h |
| Tue Mar 08 10:04:54 CET 2005 | Started checking host 'kharga.cepba.upc.es'. |

OK Cancel

ons

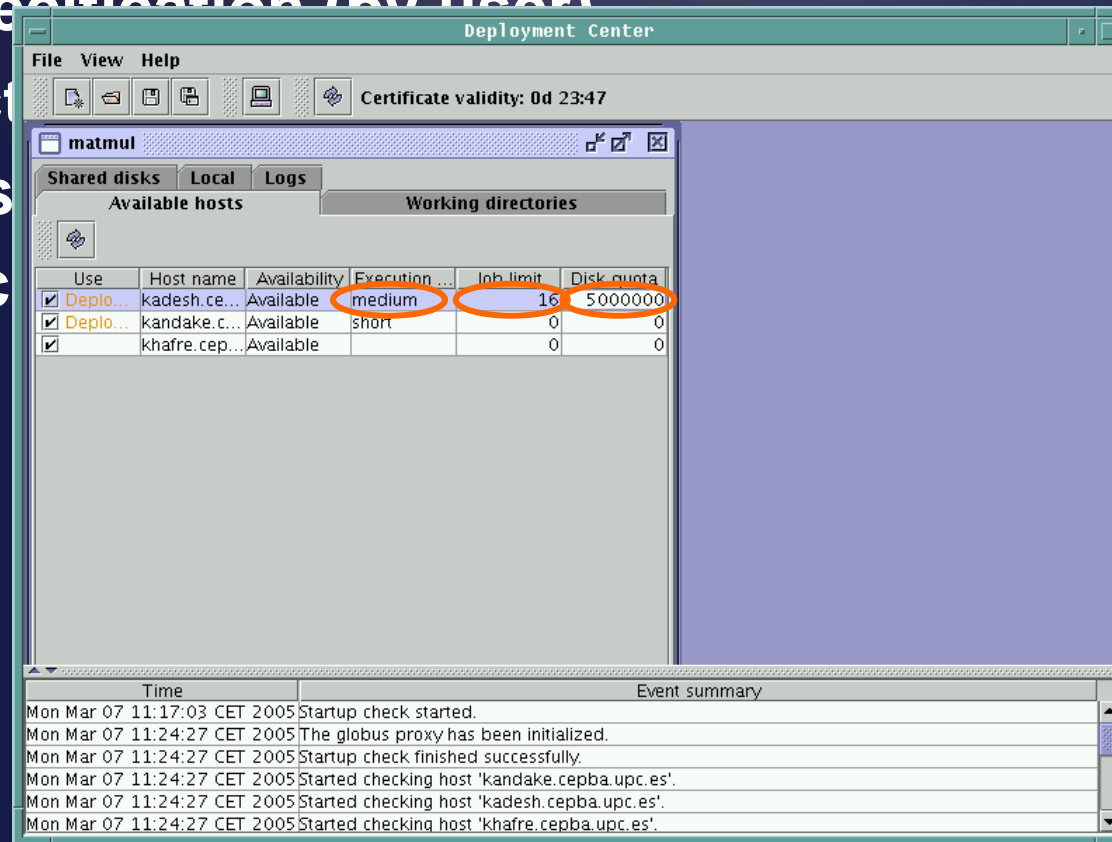
3. Deployment of GRID superscalar applications

Project specification (by user)

– Select

Afterwards

A project c



4. Runtime resource selection

Runtime evaluation of the functions

- Constraints and performance cost functions dynamically drive the resource selection

When an instance of a function is ready for execution

- Constraint function is evaluated using ClassAdd library to match resource ClassAdds with task ClassAdds
- Performance cost function used to estimate the elapsed time of the function (ET)

4. Runtime resource selection

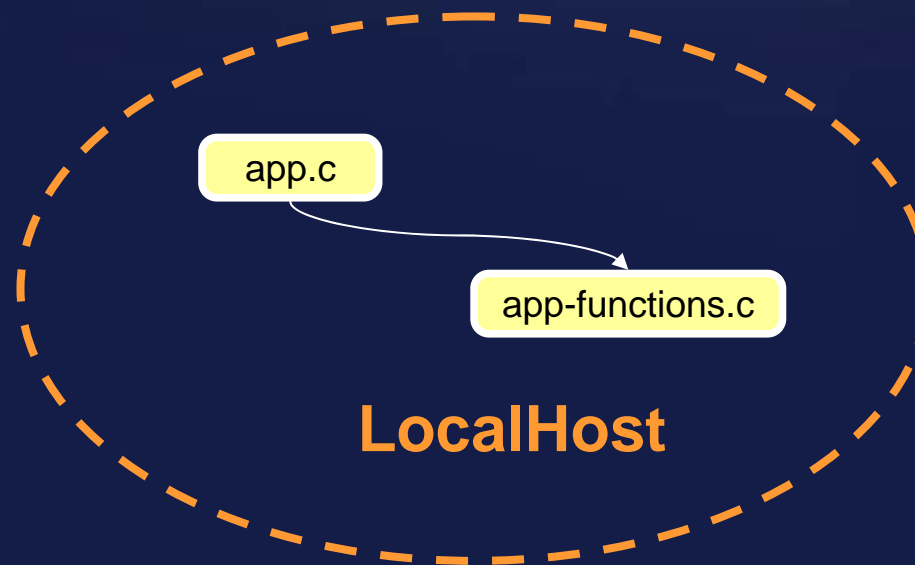
For those resources r that meet constraints

$$f(r) = \alpha \cdot FT(r) + \beta \cdot ET(r)$$

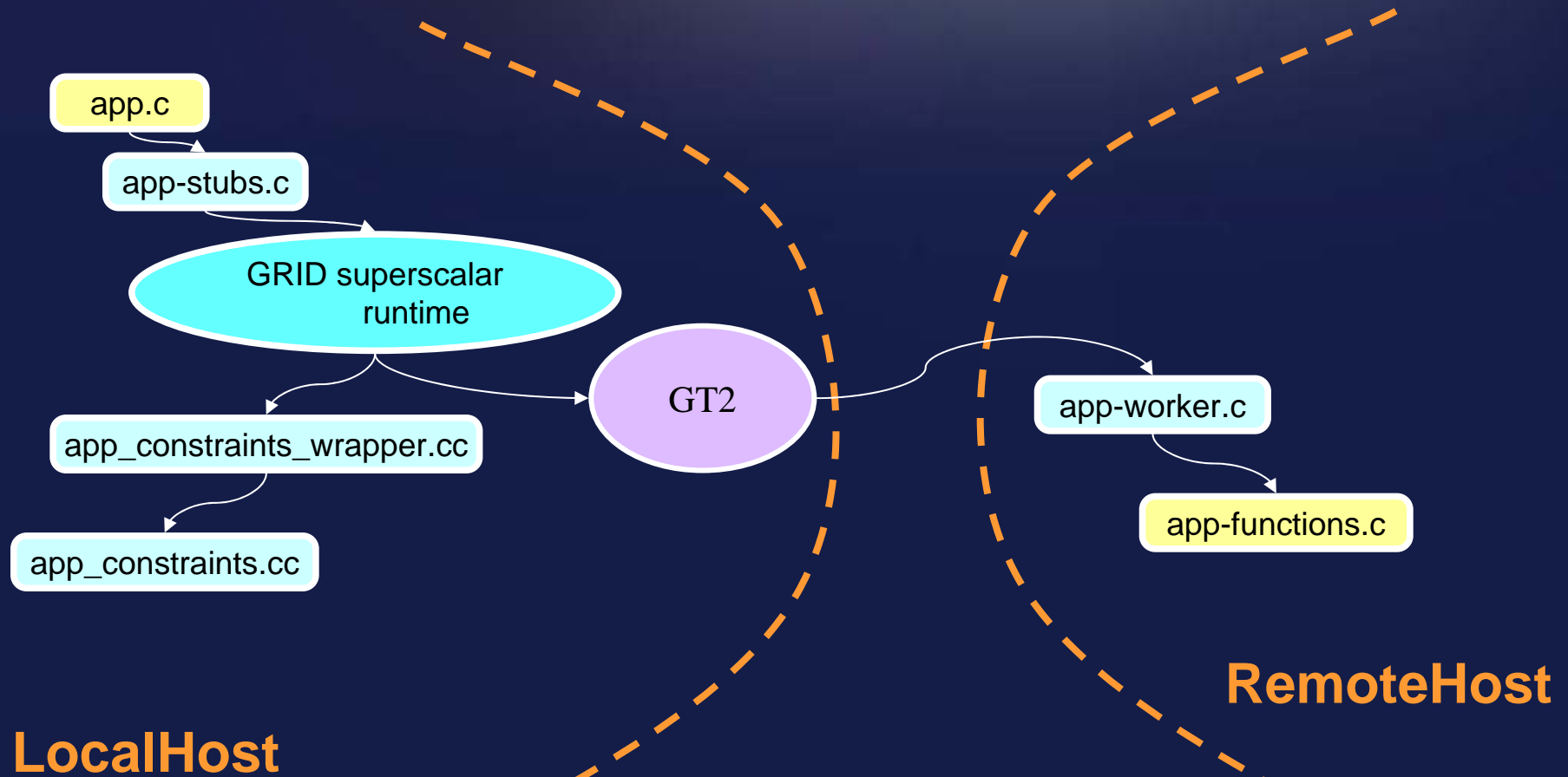
FT= File transfer time to resource r

ET = Execution time of task on resource r

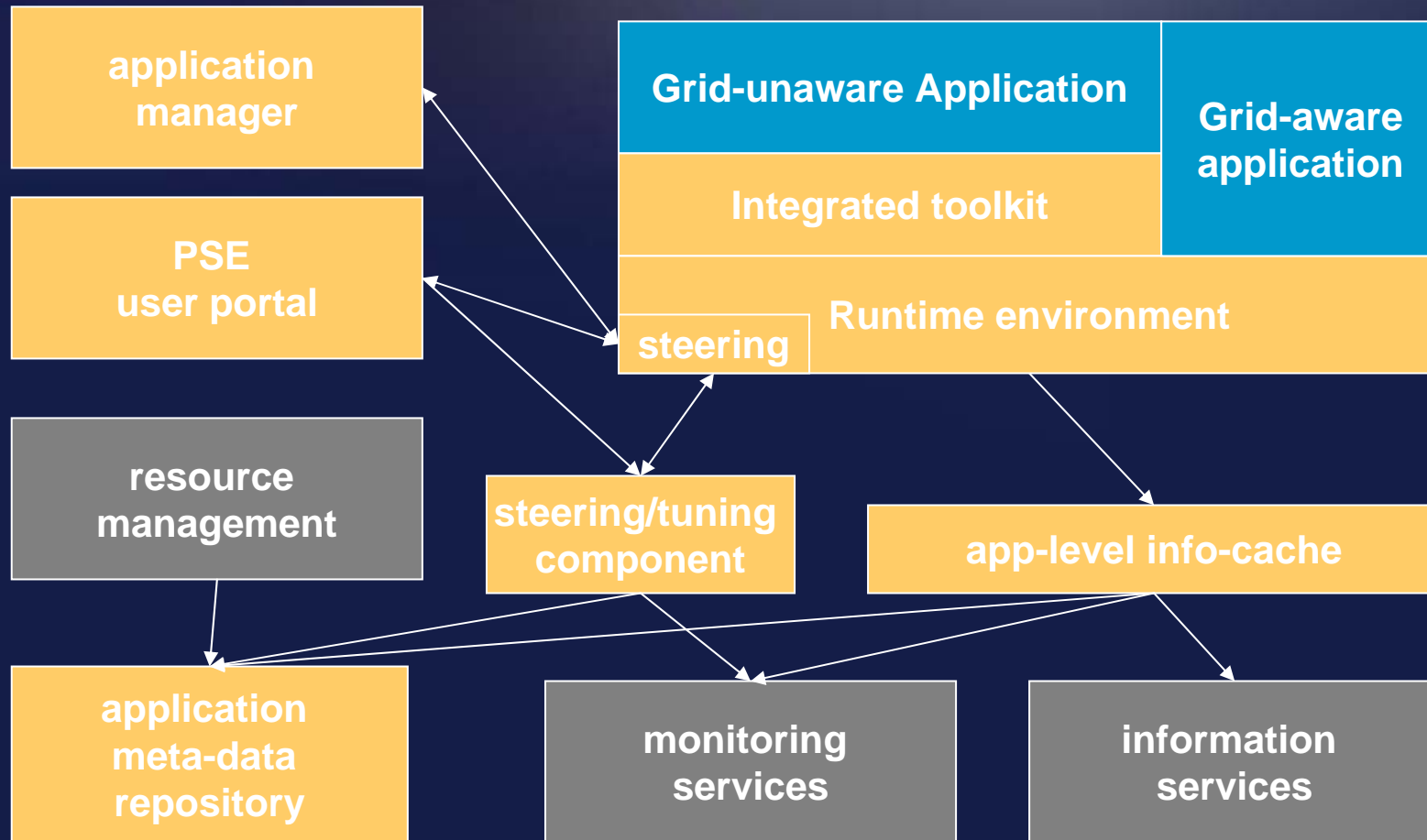
4. Runtime resource selection: call sequence



4. Runtime resource selection: call sequence



5. Plans for CoreGRID WP7



5. Plans for CoreGRID task 7.3

Leader: UPC

**Participants: INRIA, USTUTT, UOW, UPC, VUA,
CYFRONET**

- **Objectives:**
 - **Specification and development of an
Integrated Toolkit for the Generic platform**

5. Plans for CoreGRID task 7.3

The integrated toolkit will

- provide means for simplifying the development of Grid applications
- allow executing the applications in the Grid in a transparent way
- optimize the performance of the application

5. Plans for CoreGRID task 7.3: subtasks

Design of a component oriented integrated toolkit

- Applications basic requirements will be mapped to components – based on the generic platform (task 7.1)

Definition of the interface and requirements with the mediator components

- Tightly performed with the definition of the mediator components (task 7.2)

Component communication mechanisms

- Enhancement of the communication of integrated toolkit application components

5. Plans for CoreGRID task 7.3

Ongoing work

- Study of partners projects
- Definition of Roadmap
- Integration PACX-MPI Configuration manager with GRID superscalar deployment center
- Specification of GRID superscalar based on the component model

Summary

GRID superscalar has proven to be a good solution for programming grid-unaware applications

New enhancements for resource selection are very promising

Examples of extensions

- **Cost driven resource selection**
- **Limitation of data movement (confidentiality preservation)**

Ongoing work in CoreGRID to integrate with component-based platforms and other partners tools