DIANE DIstributed ANalysis Environment

WeiLong UENG
ASGC
wlueng@twgrid.org

Outline

- History
- Motivation
- Overview
- DIANE Architecture
- DIANE Computing Model
- DIANE Application Framework
- Use Case - Application Implementation

History

- **The DIANE R&D Project was started at CERN in 2000**
  - Initially it was intended to be specific to the distributed data analysis for High Energy Physics
  - Later the scope was extended and the tool was sucessfully used for Monte Carlo simulations based on Geant4 toolkit.
- **DIANE gained popularity in various user communities as EGEE project involved more scientific communities.**

Motivation

- **Using middleware directly requires a lot of manual work**
- **Not easy to monitor the job progress and cancel jobs using middleware**
- **Based on users' requirement**
  - *Integration of task results*
  - *monitoring of job progress and individual tasks*
  - *automatic error-recovery policies*
  - *granularity of the size of the task may change independently of the number of workers -- natural load-balancing and optimization of performance*
  - *uniform, transparent and easy user interface and API hiding complexity of underlying middleware mechanisms*
  - *the same API and UI is used when running local jobs and GRID jobs*

Motivation

- **DIANE will …**
- **Help application communities and smaller Virtual Organizations using the distributed computing infrastructures more efficiently**
- **Lead to an improvement of the quality of service of the EGEE/LCG Grid**

Overview

- **A framework for efficient control and scheduling of computations on a set of distributed worker nodes.**
- **DIANE Allows you**
  - Reduce the application execution time by using the resources more efficiently,
  - Reduce the user work overhead by providing fully automatic execution and failure management,
  - Efficiently integrate local and Grid resources.

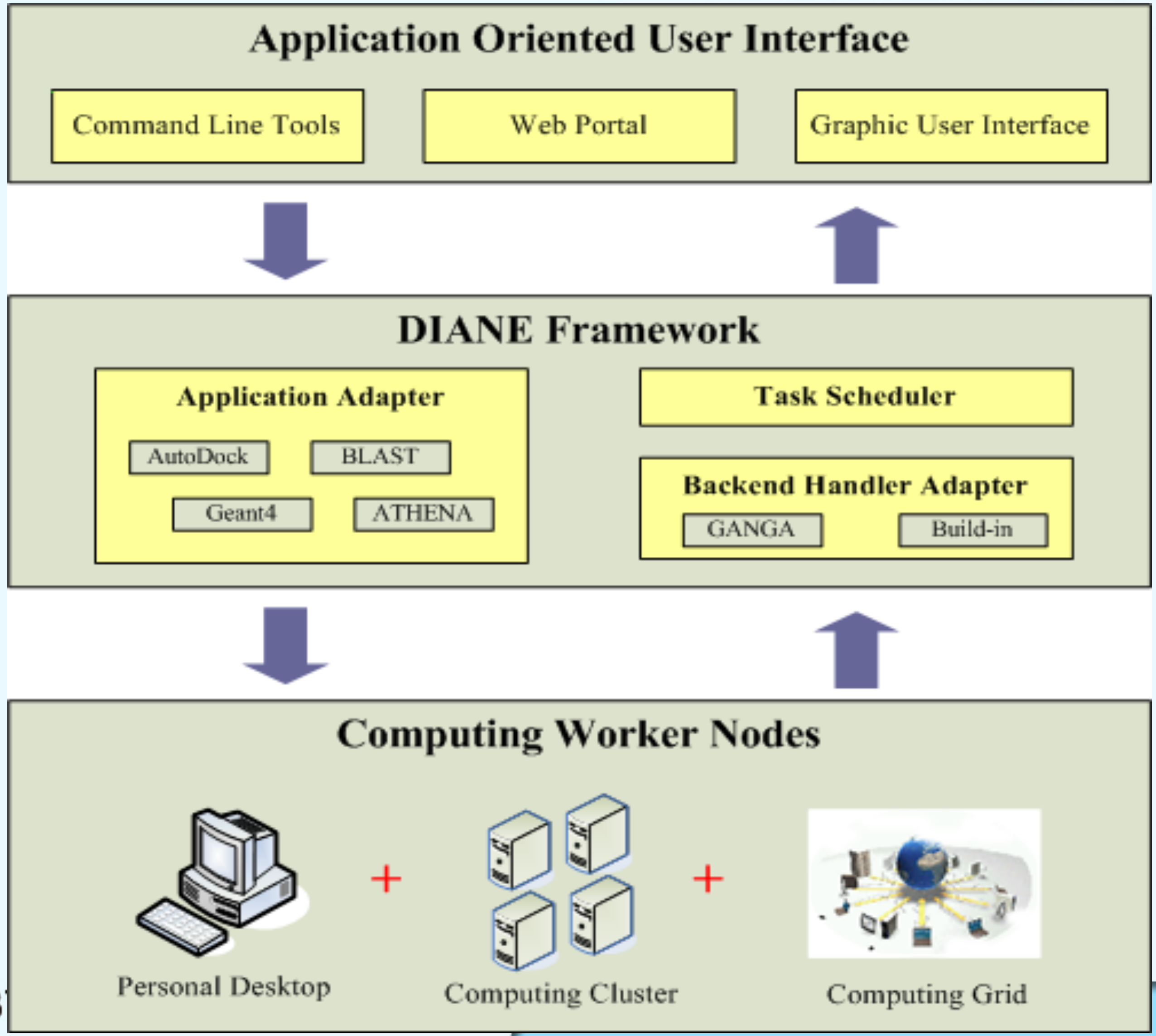
Overview

- **A Light-weight Distributed Framework**
- **Synchronization, Communication and Workload Management**
  - The execution of a job is fully controlled by the framework which decides when and where the tasks are executed.
- **A Thin Software Layer**
  - Easily work on top of more fundamental middleware, such as PBS, LSF or the Grid resources

What DIANE Used For?!

- Managing large number of small independent tasks (typically for parametric study)
- Dynamic Workload Balancing for Parallel Applications
- User-level Job Scheduling
- Failure Recovery
- Agent-based computing or Pilot jobs in which a set of worker agents controls the resources.

What DIANE Used For?!



DIANE Features

- **DIANE core framework does not depend on any concrete application (in particular any data analysis software)**
- **Implemented in Python running CORBA**
- **Supported Language for Applications**
  - C++ and python application components are supported directly
  - Application written in any language in a form of executable file (FORTRAN, Java) may also be used
- **Latest version: DIANE 2.0-beta20**

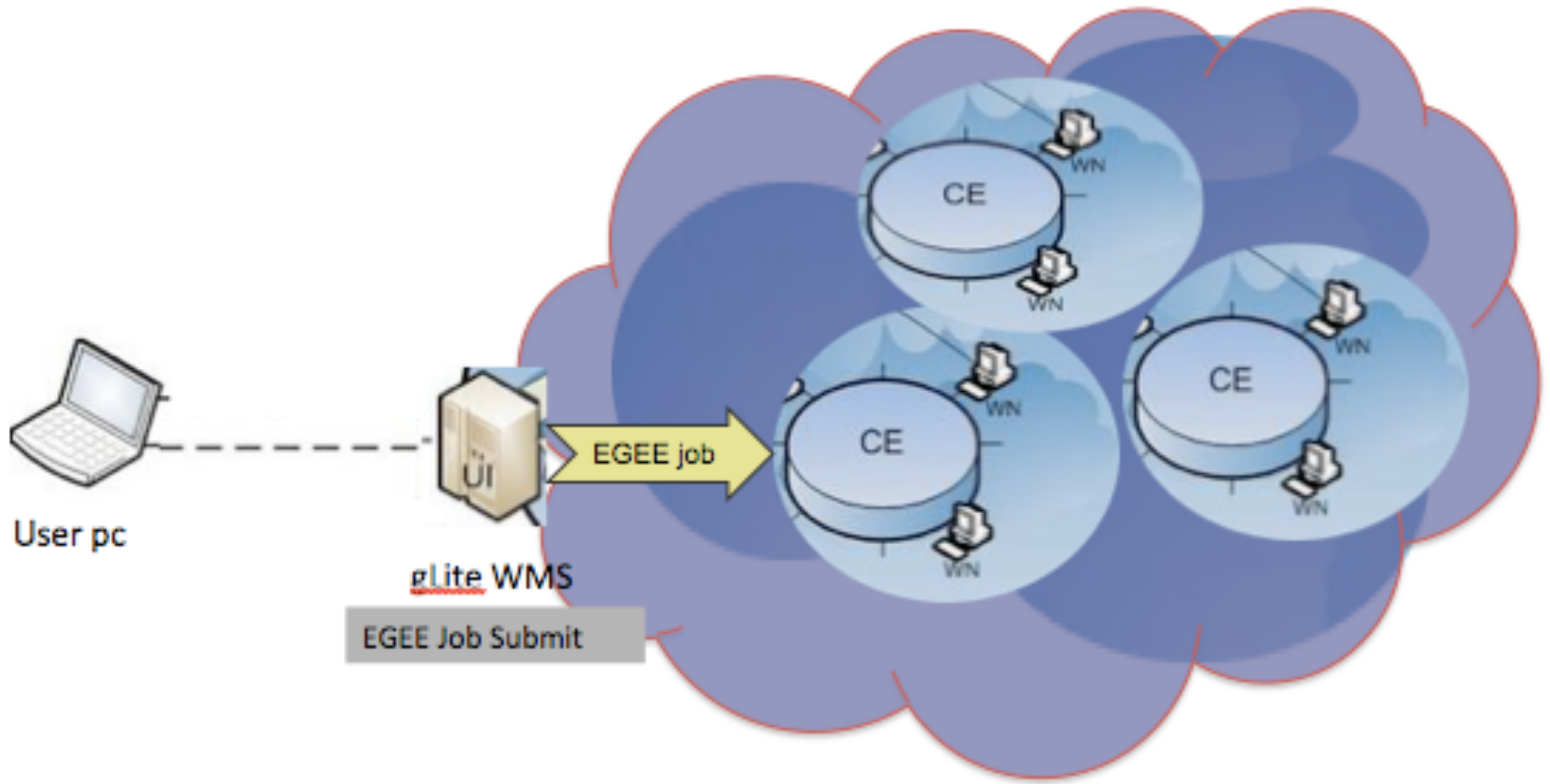
DIANE Architecture

- Handles the communication and networking transparently
- Flexibility for implementation of users’ own scheduler and worker classes to support more complex scenarios
- Completely Transparent between Applications and Backends
- Customized Error Recovery Policies

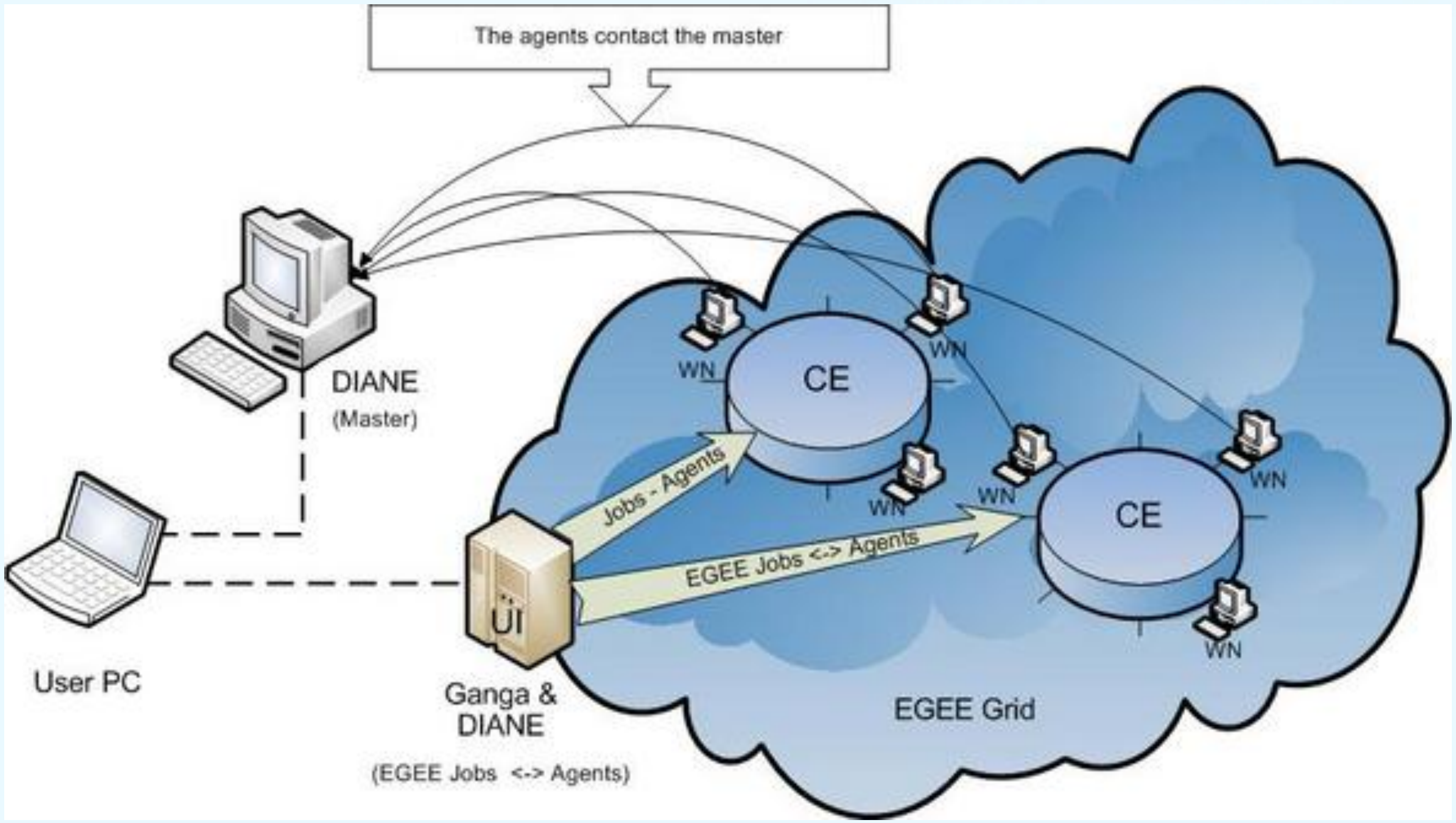
DIANE Architecture

- **Based on Master-Worker Model**
  - Improve application execution time and provide partial fault tolerance
- **Master**
  - Mapping tasks to workers and decide the policy of failures in task execution
  - Defining the application-specific action(e.g. merging of outputs)
- **Worker**
  - Tasks execution and output transmission
  - Keep contact with Master

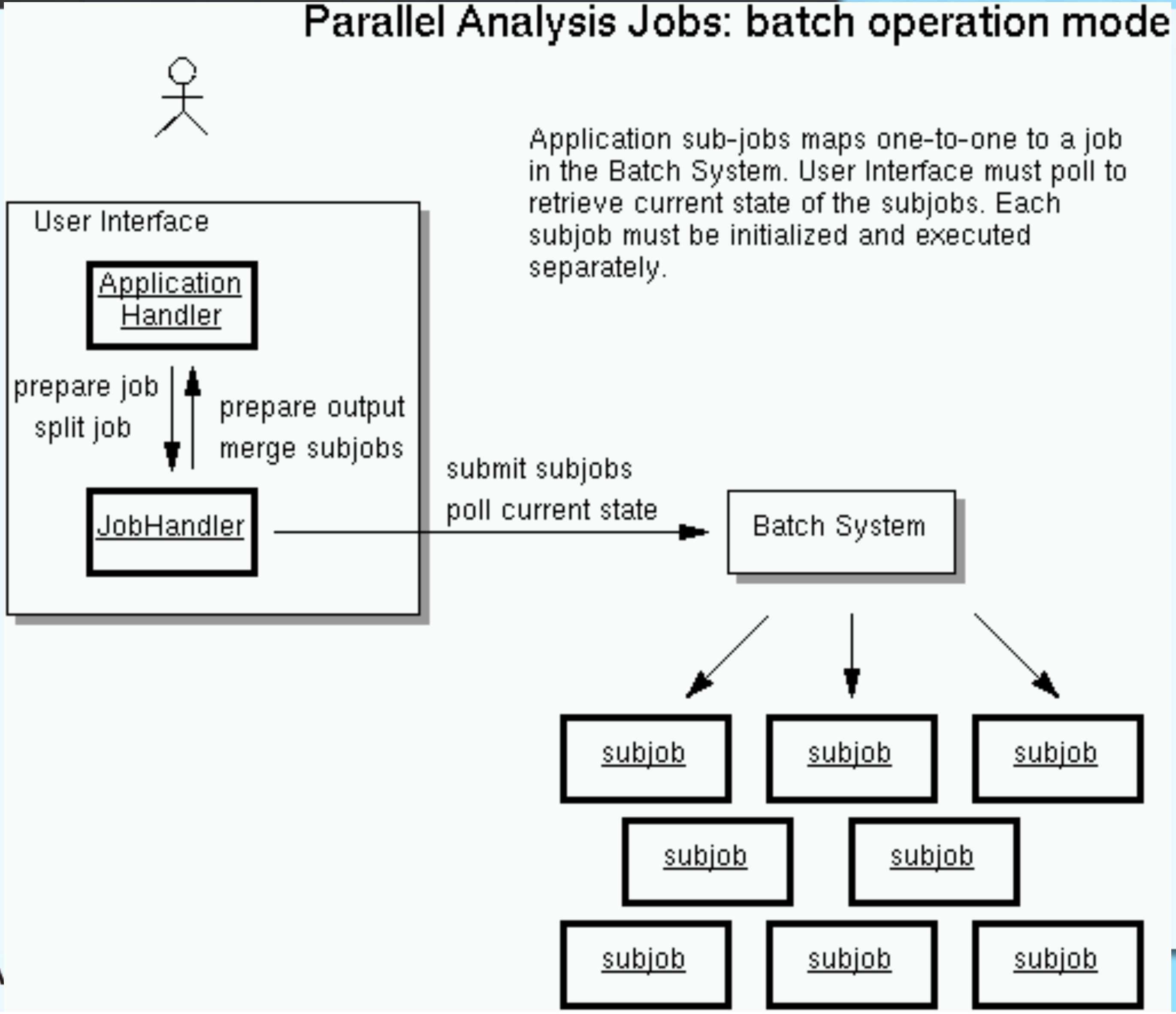
Job Submission in Grid



Job Submission using DIANE

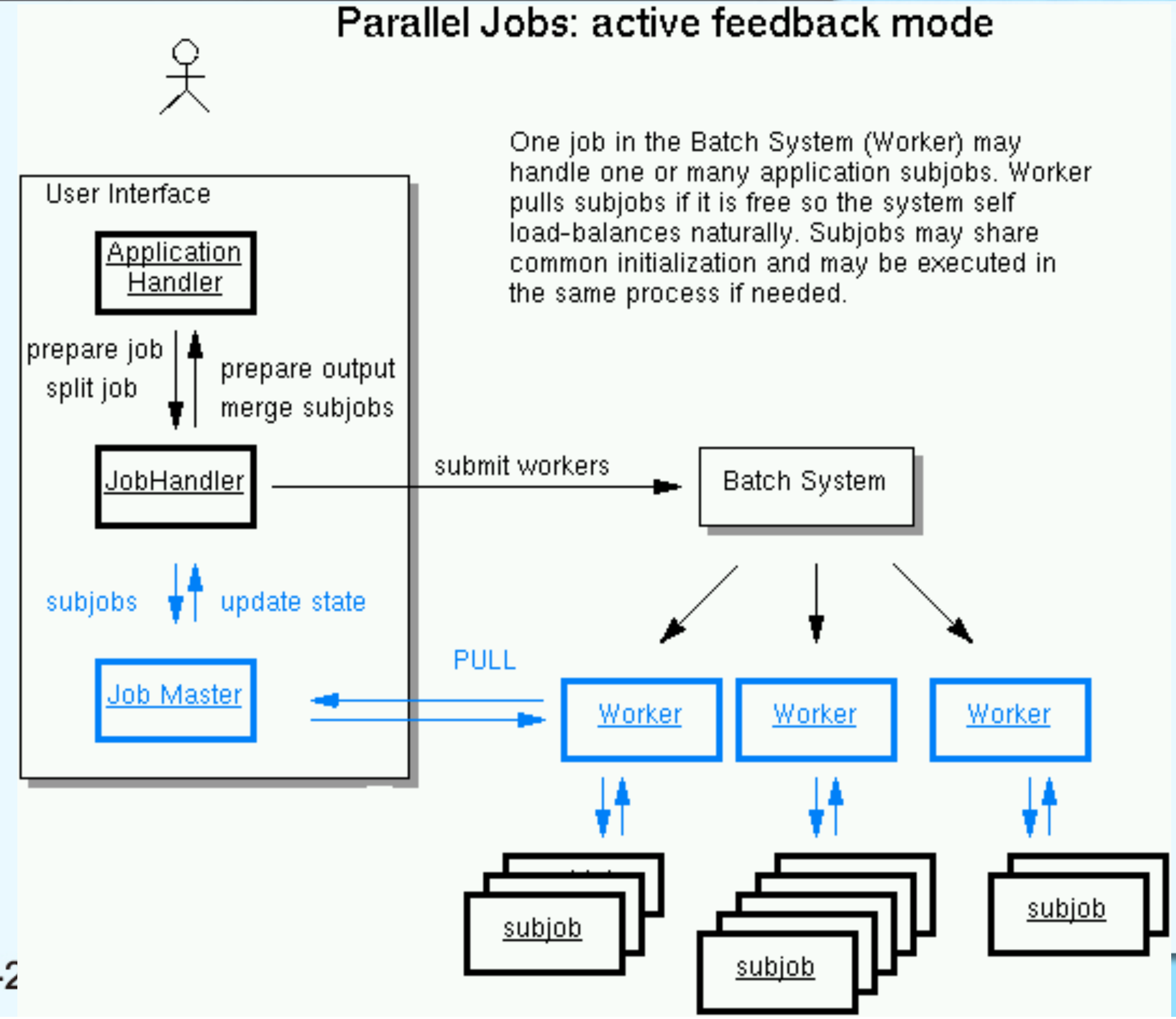


Active Feedback vs. Batch Operation Mode



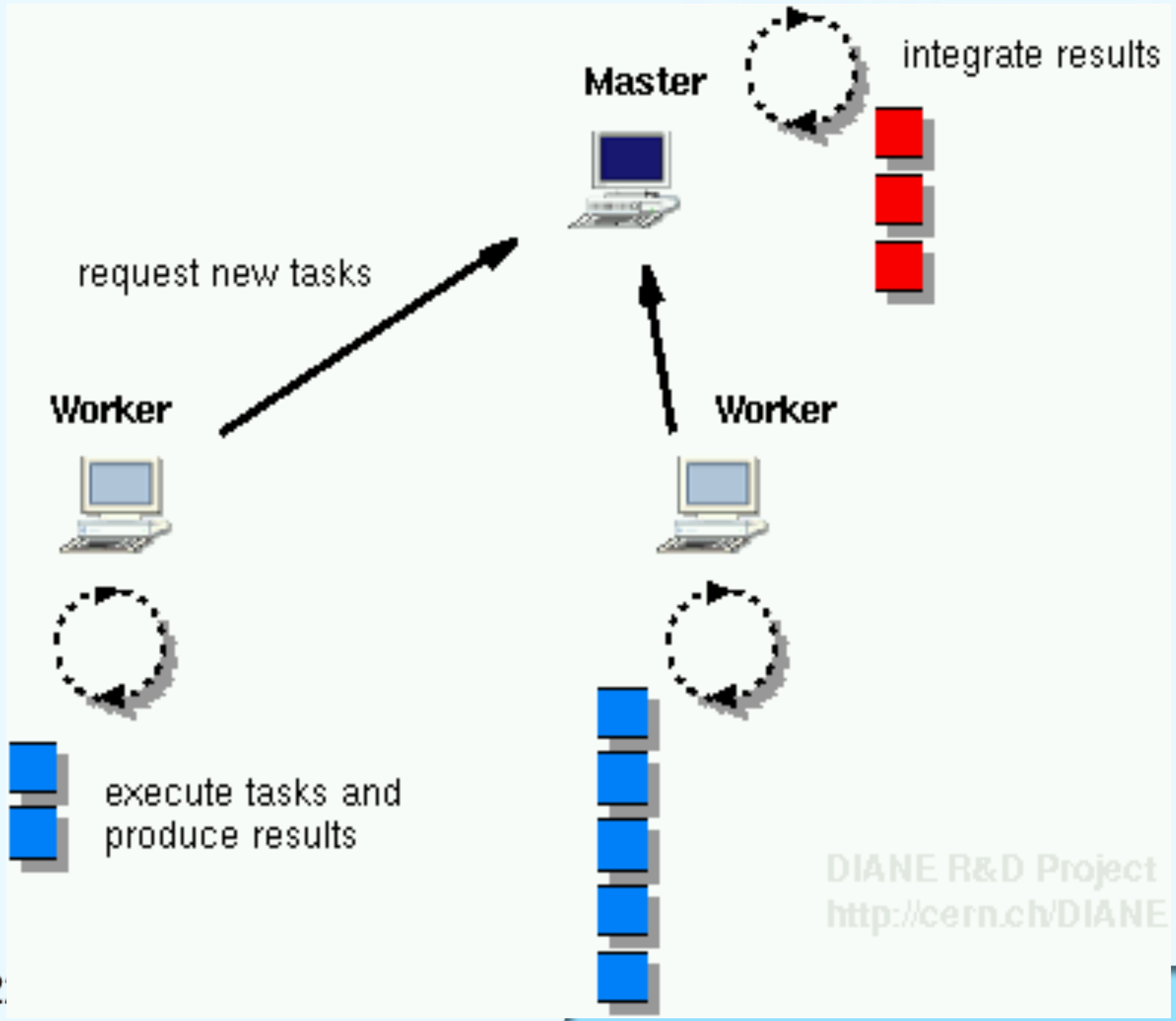
Application sub-jobs maps one-to-one to a job in the Batch System. User Interface must poll to retrieve current state of the subjobs. Each subjob must be initialized and executed separately.

Active Feedback vs. Batch Operation Mode



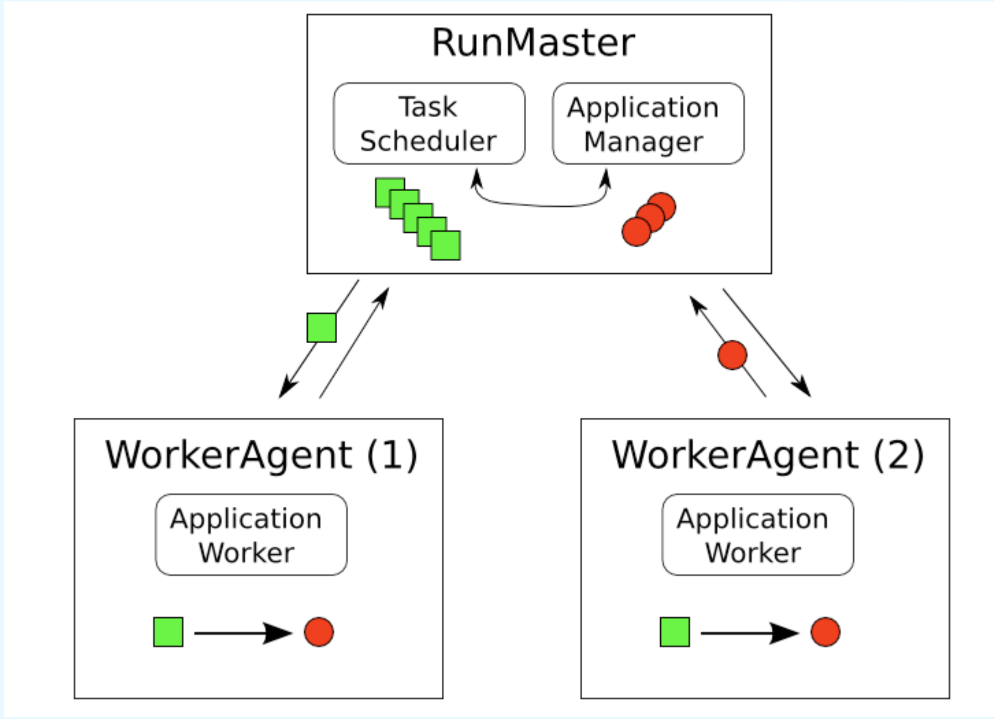
One job in the Batch System (Worker) may handle one or many application subjobs. Worker pulls subjobs if it is free so the system self load-balances naturally. Subjobs may share common initialization and may be executed in the same process if needed.

DIANE Computing Model

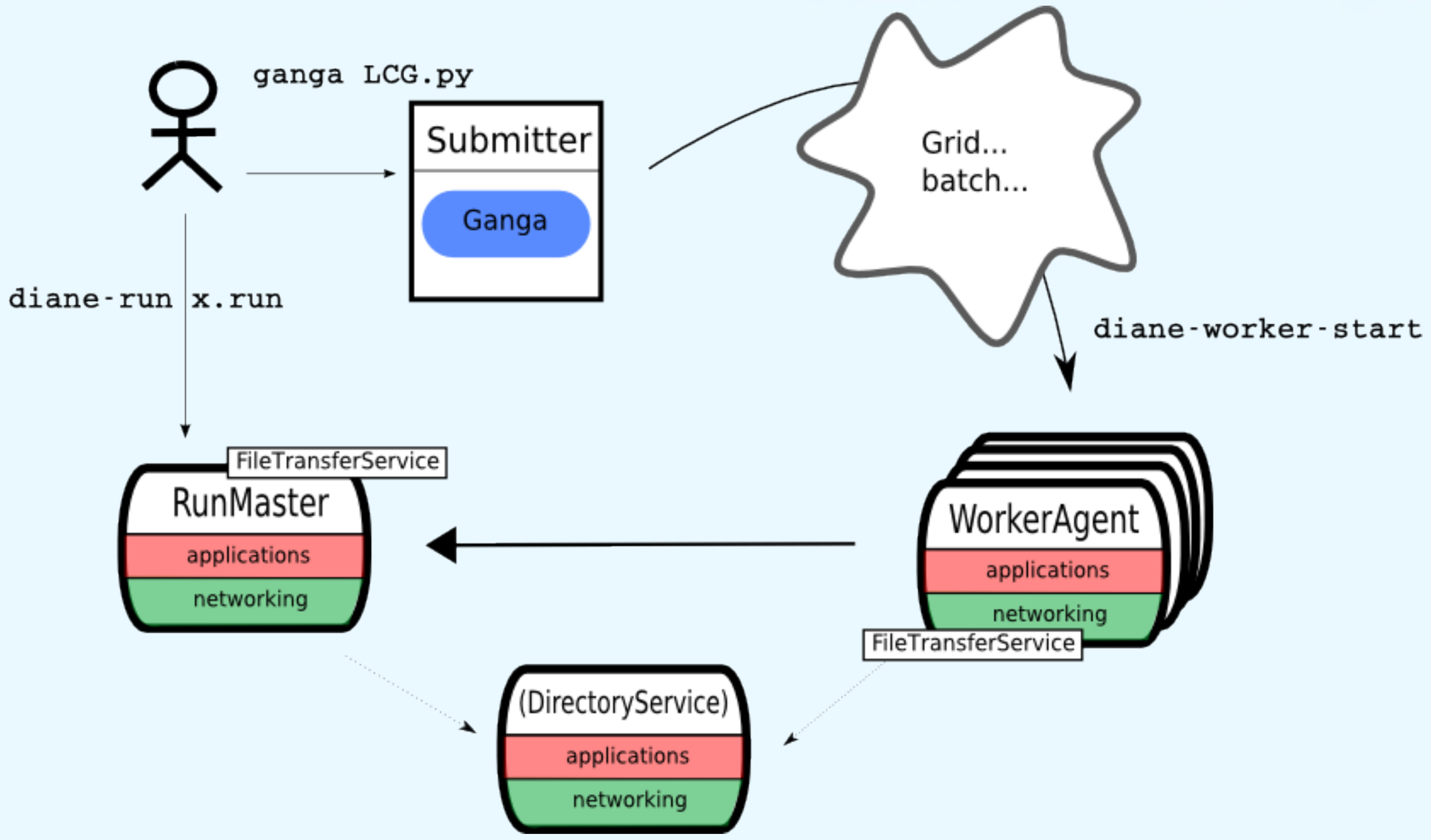


DIANE Application Framework

- **on the master node:**
  - Task Scheduler
  - Application Manager
  - Run Manager
- **on the worker nodes:**
  - Application Worker
  - Worker Agent



DIANE Architecture

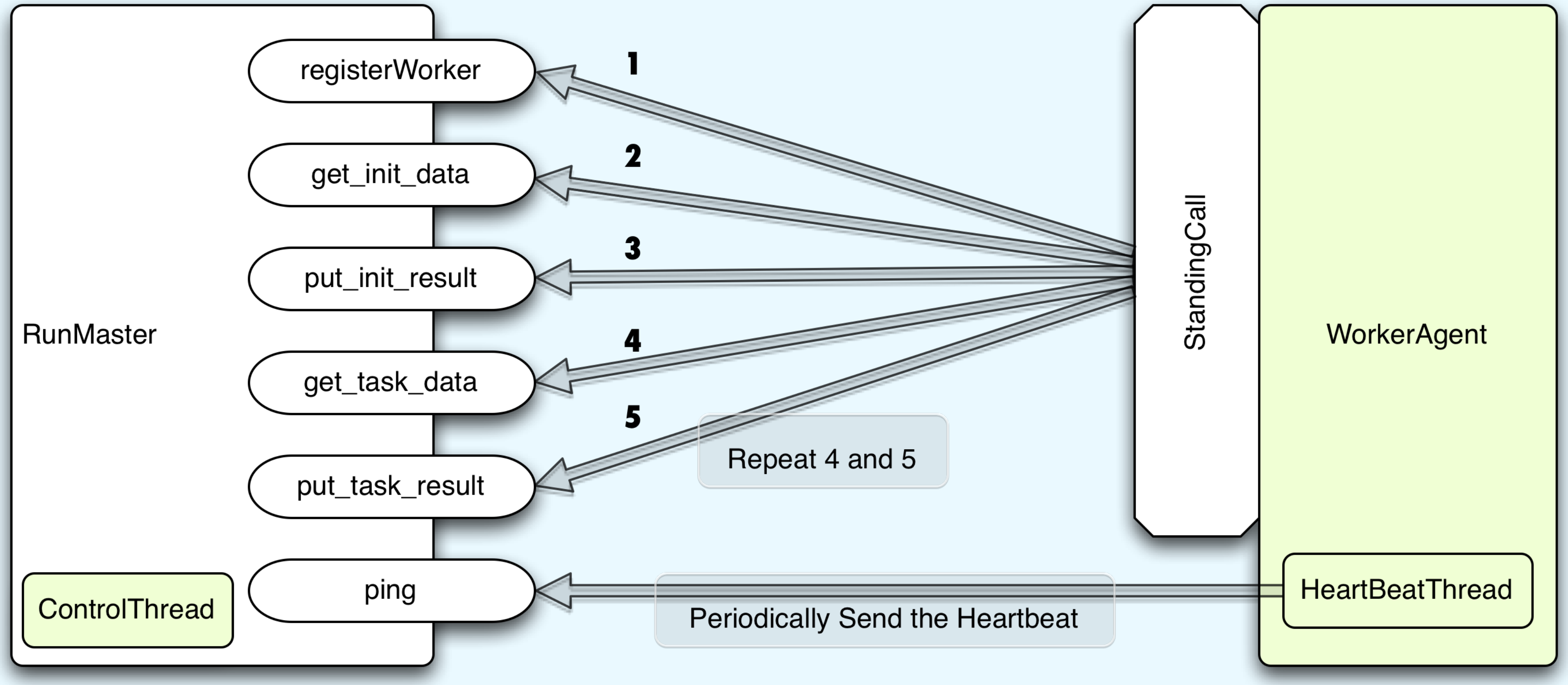


DIANE Framework
Application Components

- **They are python classes implement the following interfaces**
  - on the master node:
    - task scheduler: diane.ITaskScheduler.ITaskScheduler
    - application manager: diane.IApplicationManager.IApplicationManager
  - on the worker nodes:
    - application worker: diane.IApplicationWorker.IApplicationWorker

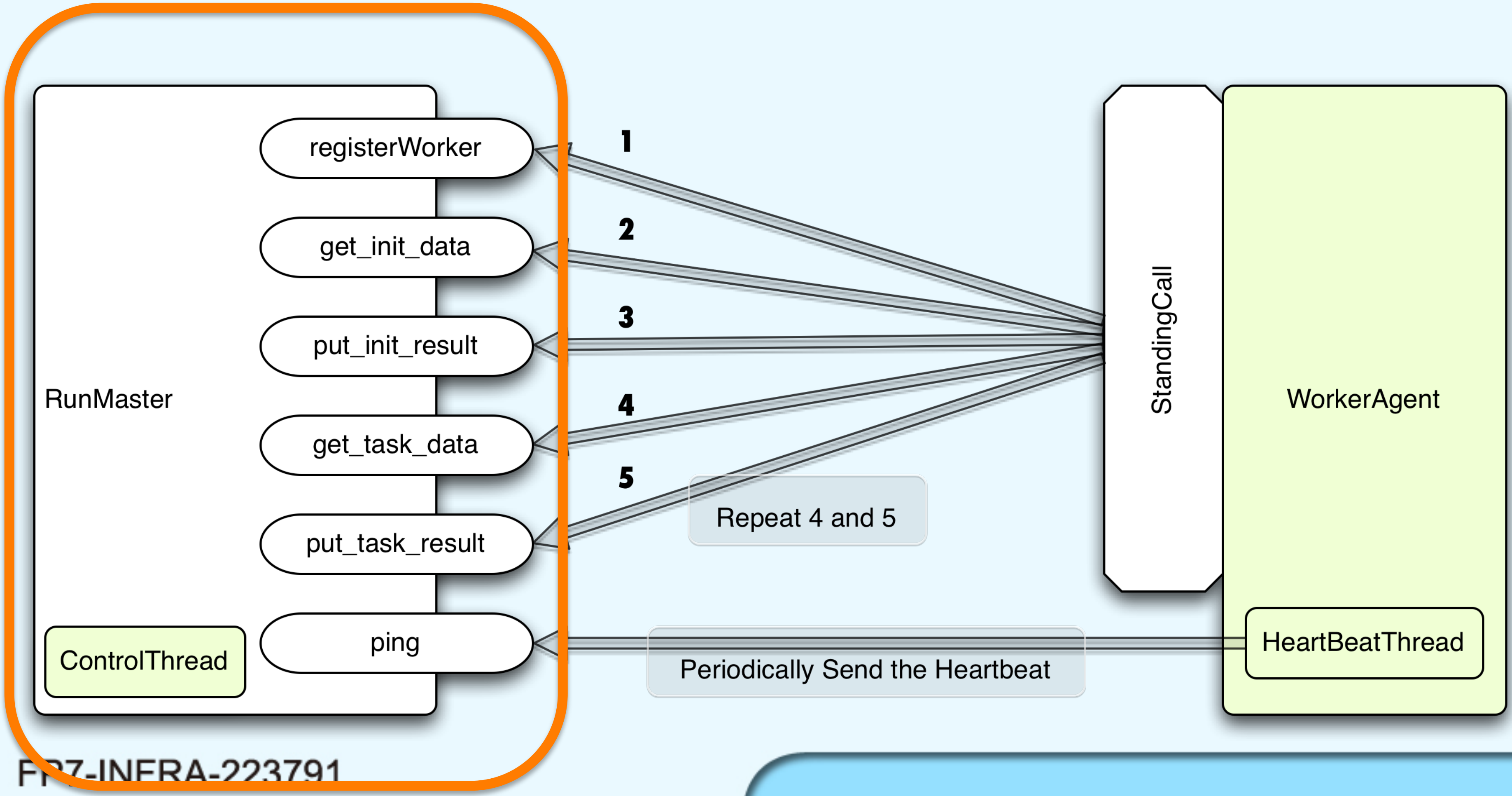
DIANE Framework

- **Communication between Master and Worker**



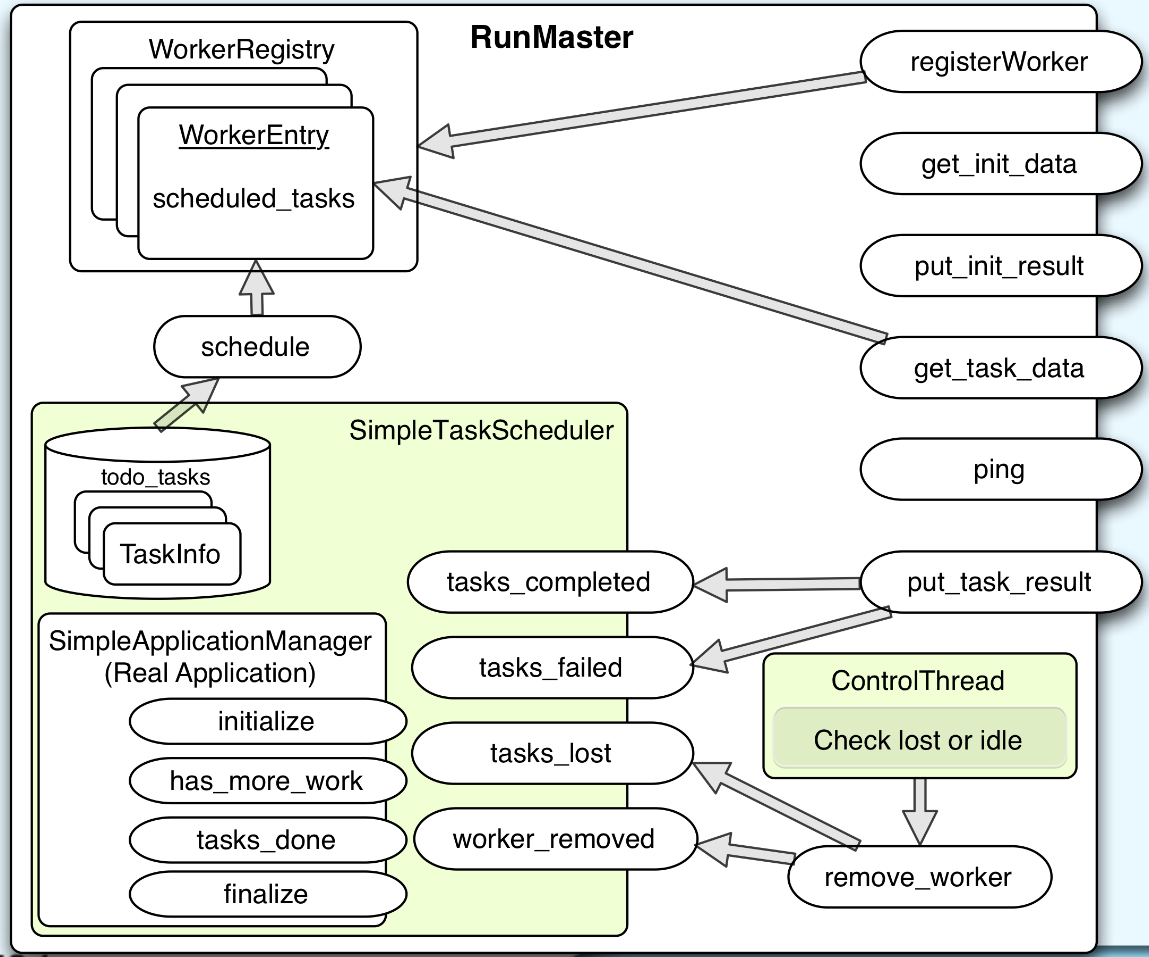
DIANE Framework

- **Look at Master side**



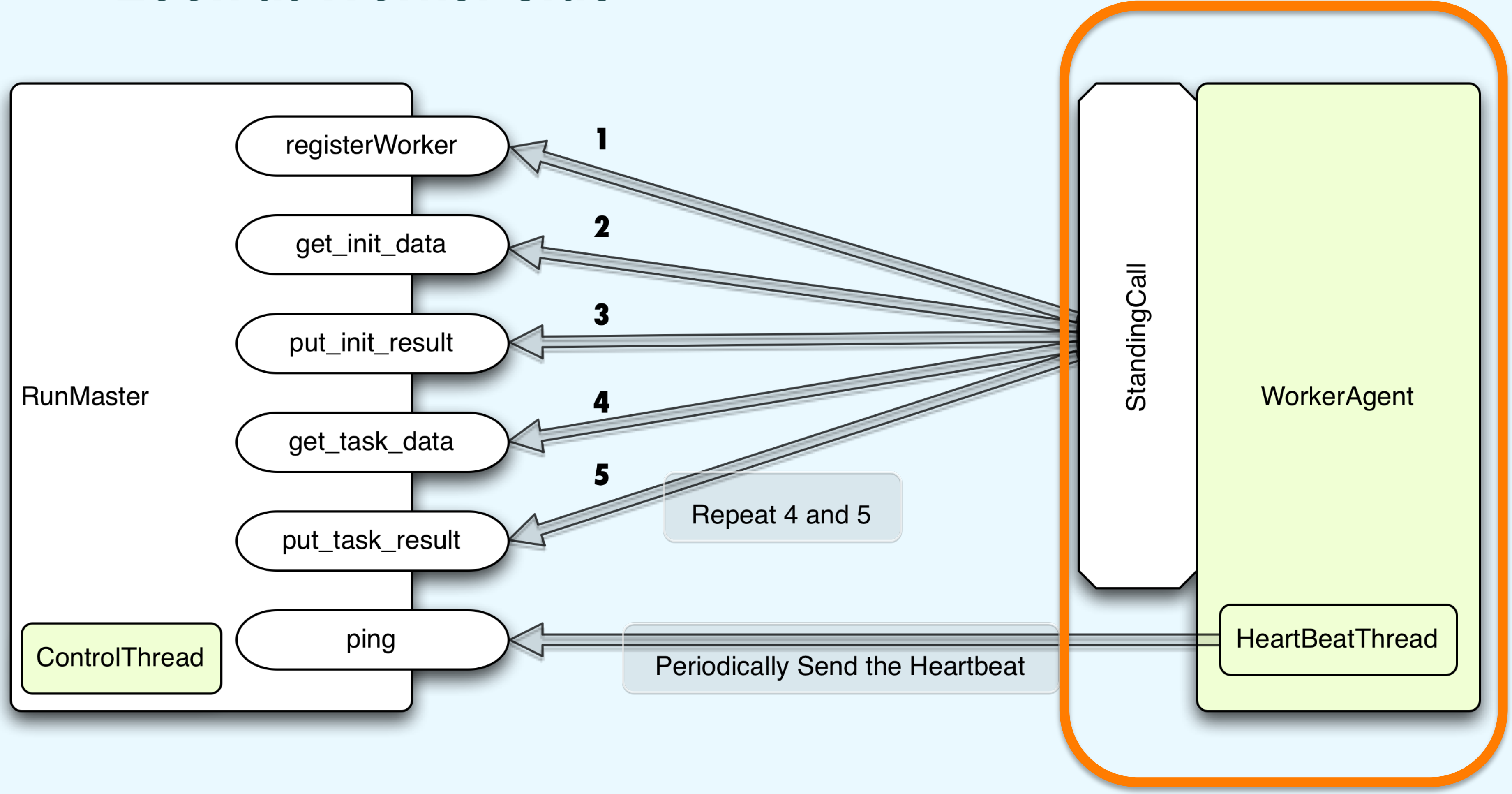
DIANE Framework

- **The Internal of Master(RunMaster)**



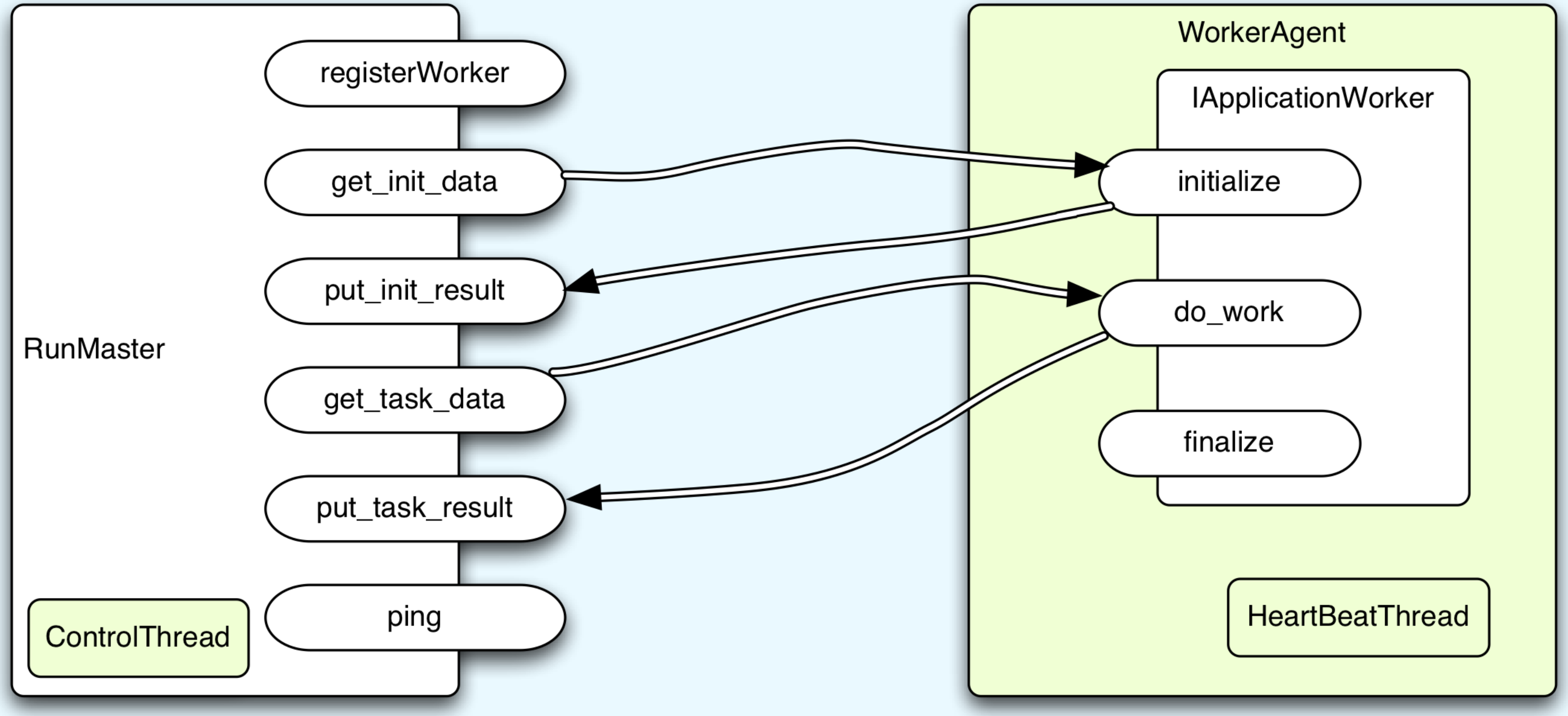
DIANE Framework

- **Look at Worker side**



DIANE Framework

- **The Internal of Worker Agent**

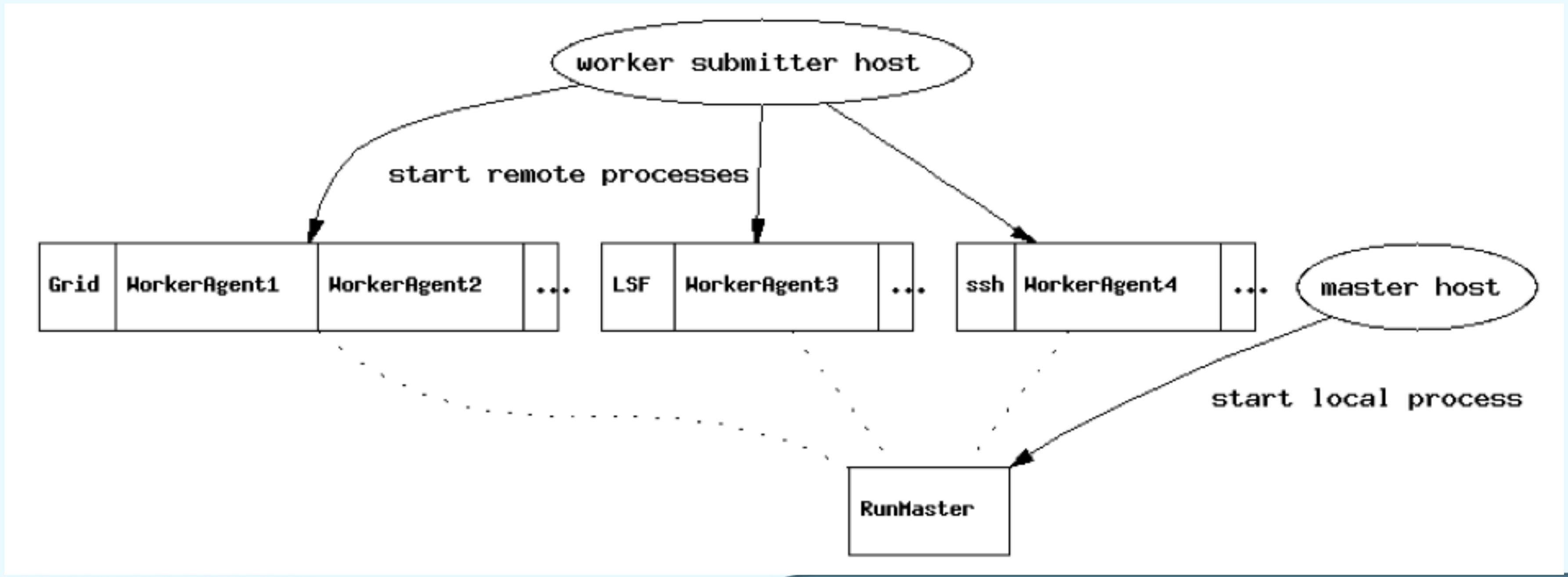


DIANE Framework
Networking and Communication Layer

- **Invisible to the application**
- **Reliable messaging between the master and the workers**
- **Always unidirectional**
  - From the workers to the master
  - Allows the control of the worker agents by the master as if the communication was bidirectional

DIANE Framework

- **diane-worker-start**
- **Use Ganga as workerAgent jobs submitter**



DIANE - Example

Here’s the example of simple executable application

Hello script

```
#!/usr/bin/env bash
rm -f message.out
echo hello $* > message.out
echo "I said hello $* and saved it in message.out"
```

Suppose to run this script for 20 times the “Hello” executable script, changing its arguments every time

DIANE - Example

File: hello.run (it’s a simple python file defining the work to be done)

```
# tell DIANE that we are just running executables
# the ExecutableApplication module is a standard DIANE test application

from diane_test_applications import ExecutableApplication as application

# the run function is called when the master is started
# input.data stands for run parameters
def run(input,config):
        d = input.data.task_defaults # this is just a convenience shortcut

        # all tasks will share the default parameters (unless set otherwise in individual task)
        d.input_files = ['hello']
        d.output_files = ['message.out']
        d.executable = 'hello'

        # here are tasks differing by arguments to the executable
        for i in range(20):
                t = input.data.newTask()
                t.args = [str(i)]
```

Now you can start the master using the run file:

$ diane-run hello.run

DIANE - Example

• The master will start in its own run directory
(this information is printed by the master - check the output).

• The rundir is typically located in ~/diane/runs/nnn.
• The default location may be changed with $DIANE_USER_WORKSPACE environment variable.

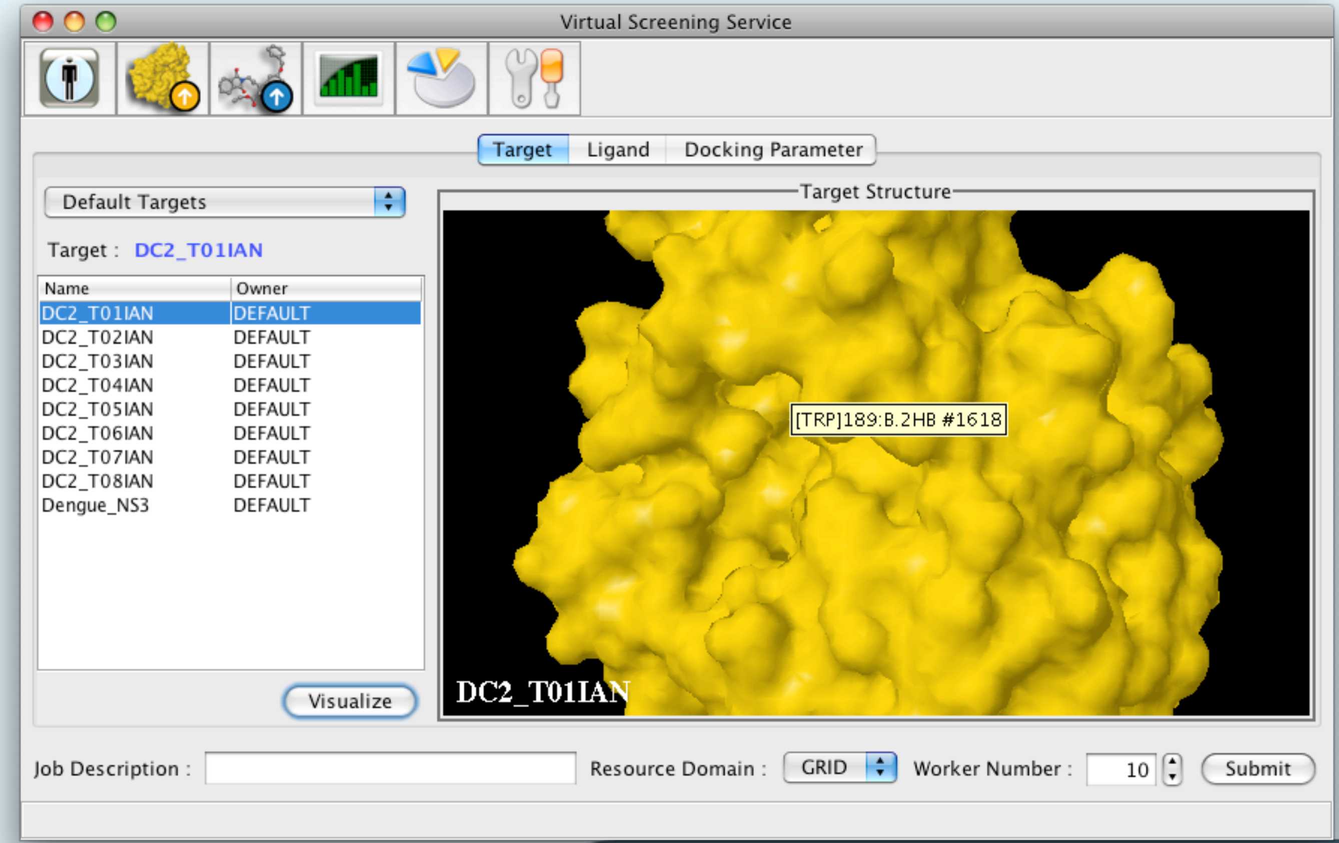
To start a couple of worker agents:

```
$ ganga LocalSubmitter.py --diane-worker-number=5
```

• All results are stored by the master in the run directory
(this behaviour may be customized and depends on the application plugins).

Use Case - Application Implementation

- **Drug Discovery Application**
  - Developed by ASGC, TW

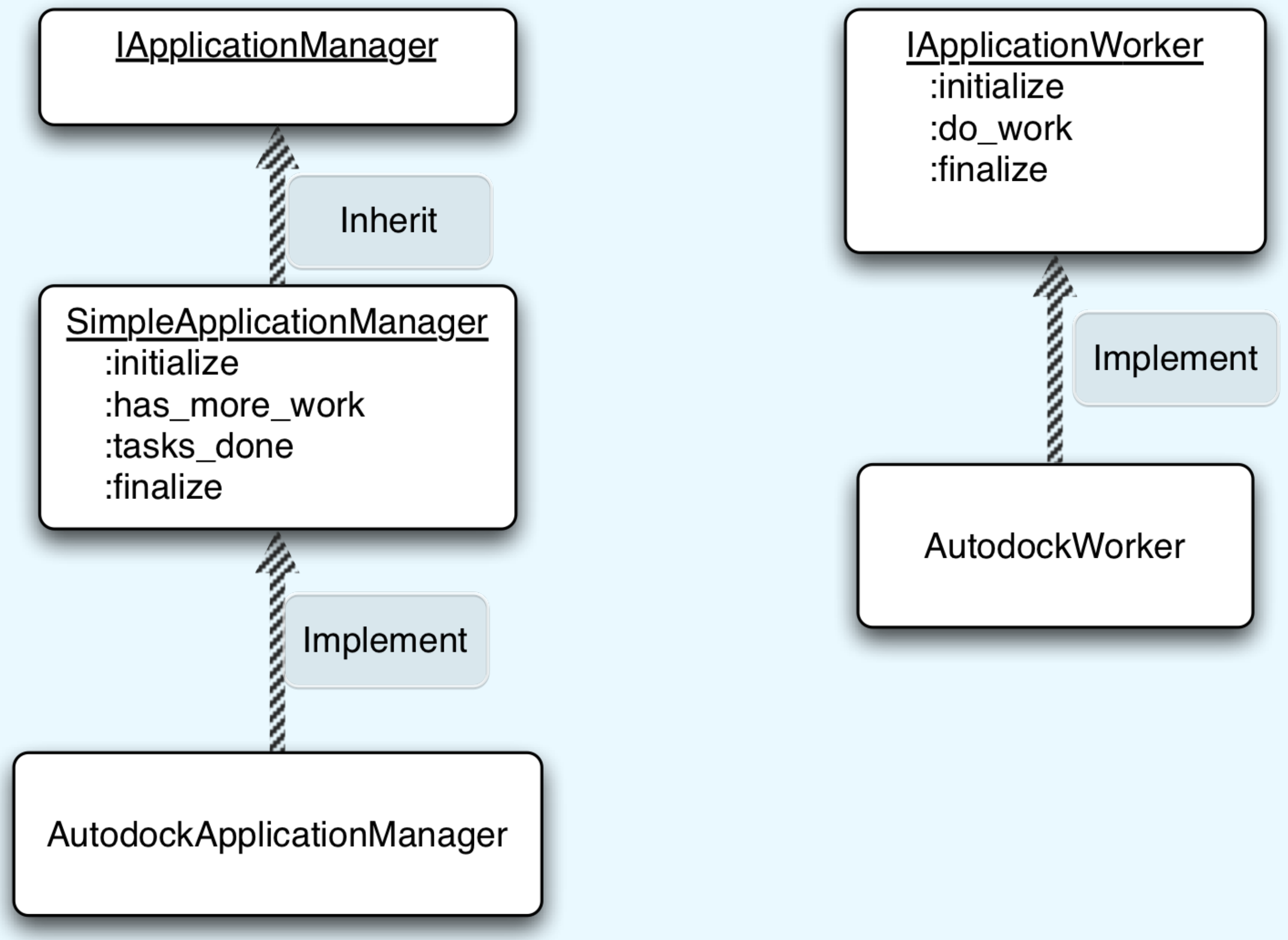


Target : DC2_T01IAN

Name	Owner
DC2_T01IAN	DEFAULT
DC2_T02IAN	DEFAULT
DC2_T03IAN	DEFAULT
DC2_T04IAN	DEFAULT
DC2_T05IAN	DEFAULT
DC2_T06IAN	DEFAULT
DC2_T07IAN	DEFAULT
DC2_T08IAN	DEFAULT
Dengue_NS3	DEFAULT

Job Description : Resource Domain : GRID Worker Number : 10

Drug Discovery Application Implementation



Drug Discovery Application Implementation

EUAsiaGrid

- **The internal of AutodockApplicationManager**

:initialize

1. Initialize this target and ligands by querying AMGA metadata
2. Initialize the tasks QUEUE

:has_more_work

1. Return true if the tasks QUEUE is not EMPTY, otherwise false.

:tasks_done

1. Receive the task results, and pass the results to the integration function.
2. Update the post processing results to AMGA metadata.

:finalize

1. Finalize this docking RUN.

Drug Discovery Application Implementation

- **The internal of AutodockWorker**

:initialize

1. Prepare the environment for running 'autodock'

:finalize

1. Basically do nothing

:do_work

1. Get the target and ligand information from Master(get_task_data)

2. Download the physical data of the target and ligand.

3. Run the docking simulation using autodock

4. Upload the docking result tarball to GridFTP endpoint

5. Return the docking result (put_task_result)

DIANE 2 User Interface

- **Basic Run Parameters**
  - Application
    - Define the application package (python module or package)
  - run(input, config)
    - Is a function called by diane-run which sets:
  - Input.data:
    - the application-specific input parameters
  - input.worker, input.scheduler, input.manager:
    - application control components (these parameters are typically defined in the application package and need not be specified on-per-run basis)
- **Additional run parameters**
  - diane-worker-start
    - Start a diane worker

DIANE2 User Interface - Commands

- diane-env
  - Full development environment may be specified with --devel or -d options
- e.g. >>diane-env –d bash =>source diane environment into current bash shell
- e.g. Submitting more workers which will connect to the master XXX
- >>diane-env –d ganga LCG.py –diane-run-file input.run –diane-worker-number 5 –diane-master=workspace:XXX
- diane-file-transfer
  - File transfer between diane master and worker
- diane-ls
  - List the status of masters
- diane-master-ping [kill]
  - Check if the master is alive or kill the last running master by diane-master-ping kill
- diane-run
  - Start a run of the diane master
- diane-worker-start
  - Start a diane worker

References

- **DIANE: Distributed Analysis Environment**
  – http://it-proj-diane.web.cern.ch/it-proj-diane/
- **Wiki Page**
  – https://twiki.cern.ch/twiki//bin/view/ArdaGrid/DIANE