# Ibis as Master Key

## Hands-on – advanced examples

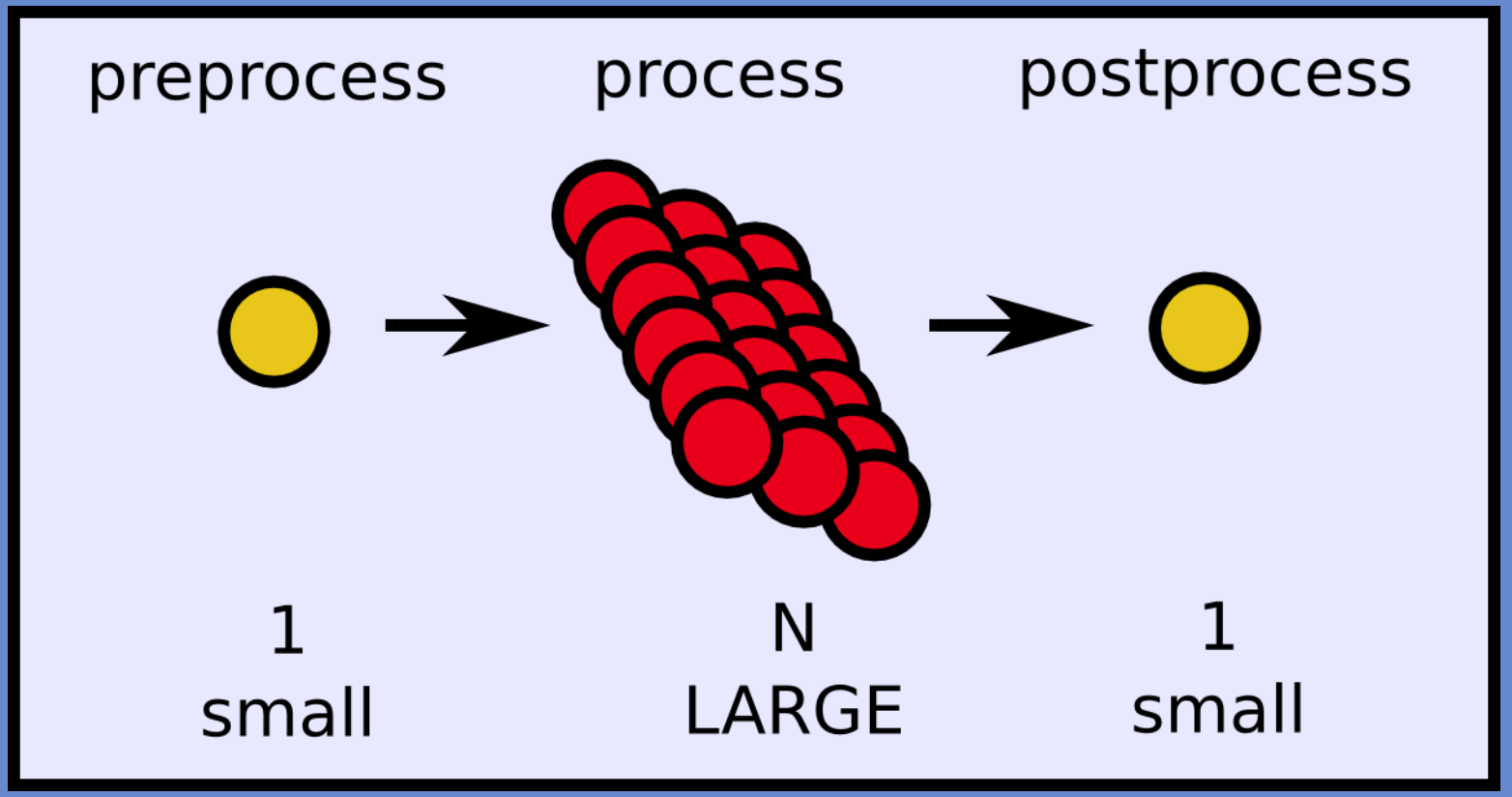
Jason Maassen

Computer Systems Group
Department of Computer Science
VU University, Amsterdam, The Netherlands

# Request
## Workflows

- Workflows (NBIC)
  - Jobs consisting of multiple stages (with dependencies)
  - Requirements vary per stage
- Ibis does not contain workflow system (yet), but:
  - JavaGAT is a perfect backend for a workflow system
    - Easy access to resources, files, etc.
- Question: how hard is it to implement a workflow ?

# Example
## Workflows



# Implementation
## Workflows

- Implementing the frontend is a lot of work
  - User friendly, flexible, GUI ?
  - Lots of systems exists!
- A JavaGAT backend is quite easy
  - We have already shown most of what you need (job submission and monitoring, file transfer, etc).
  - Still need the logic that determines what to run where and when.

# The code
## Workflows

- See masterkey.workflow package
- Main differences with previous examples:
  - Runs scripts instead of executables (more flexible)
  - Assumes data is remote
  - Reads work from a simple description file.

disclaimer: this is only a very simple example!

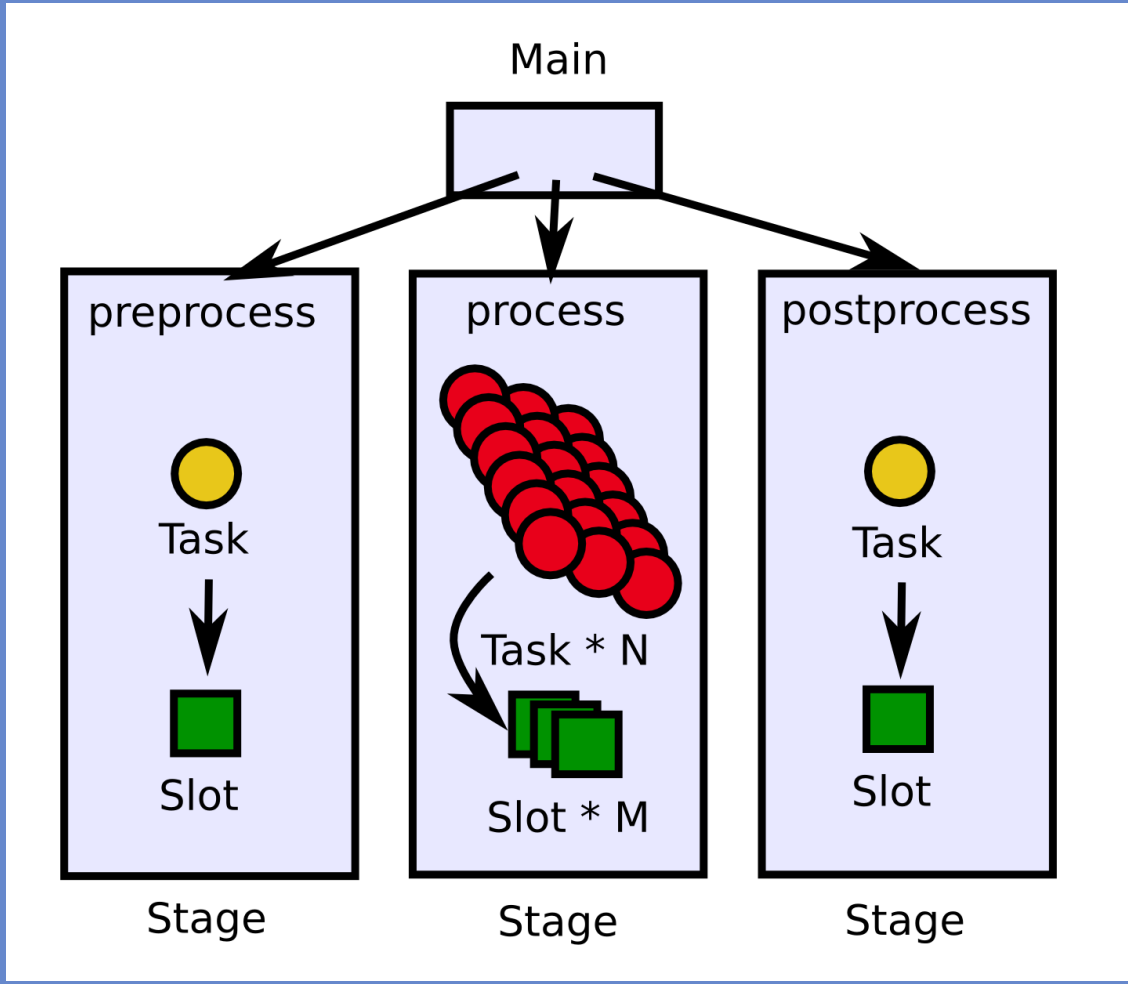
(it took about 2 hours to implement and test)

# The code
## Workflows

- Code consists of 4 classes:
  - Main is in control and creates a ...
  - Stage for each workflow stage that creates ...
  - Slots for each ‘execution slot’ and a list of ...
  - Tasks for each task to run.

# The code

## Workflows



# The code
## Workflows

- The workflow description looks like this:

```
# STAGE      RESOURCE                SLOTS  SCRIPT                 INPUT DATA              FILTER  OUTPUT DATA
#
preprocess,  ssh://fs0.das4.cs.vu.nl,    1, /home/jason/pre.sh,    /home/jason/input.tgz,      -,  /home/jason/temp/stage1
crunch,      sshpbs://fs0.das4.cs.vu.nl, 4, /home/jason/crunch.sh, /home/jason/temp/stage1, .jpg,  /home/jason/temp/stage2
postprocess, ssh://fs0.das4.cs.vu.nl,    1, /home/jason/post.sh,   /home/jason/temp/stage2,    -,  /home/jason/done.tgz
```

- Please change the input/output directories!
  - You can reuse the scripts and input file!

./scripts/run masterkey.workflow.Main example.workflow

# Requests
## Bag of tasks

- Alternative to PilotJobs
  - PilotJob framework bypasses regular scheduler (sort of)
  - PilotJob framework requires IPL/SmartSockets
  - Both make sysadmins a bit nervous! ☺
- There is an alternative way of doing this
  - JavaGAT only
  - Use only regular schedulers
  - No additional communication
  - Add multicore support

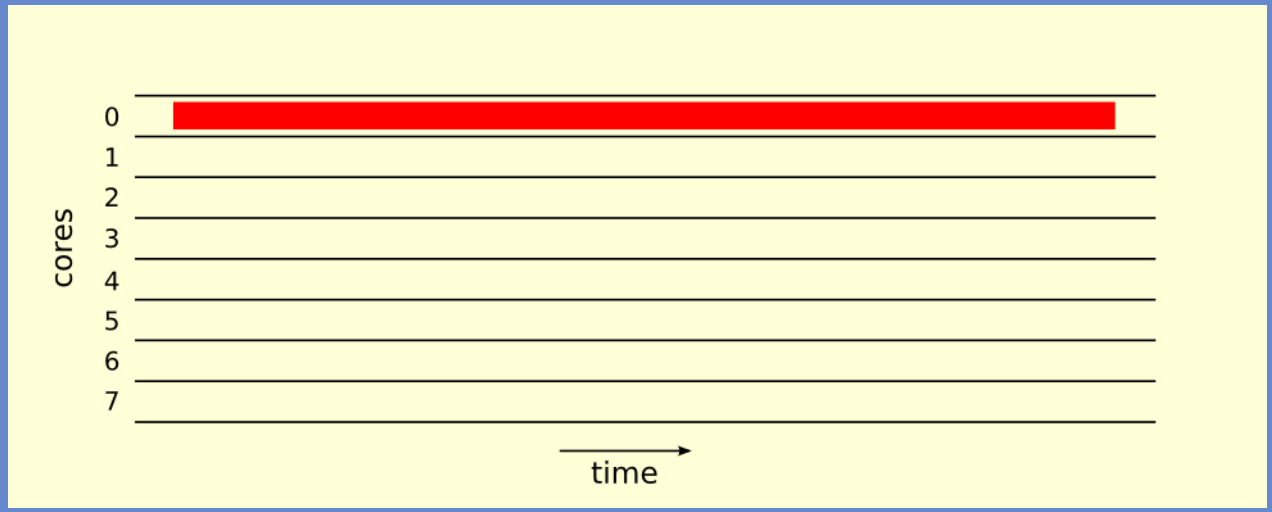
# The Problem

## Bag of tasks

- Large number of data files to be processed
  - Pick a number – any number – but make it large!
- Processing each file is sequential
  - Oops. Modern machines are multi-core and schedulers often hand out whole machines.
- Varying processing time per data file
  - Varying data sizes, processing speeds (heuristics), core speeds, etc.

# The Problem

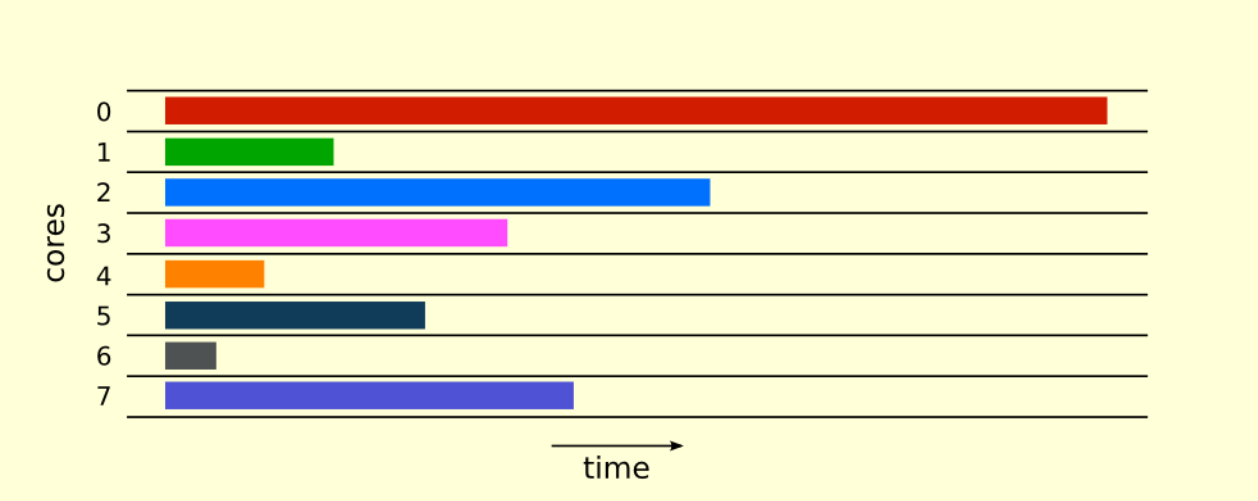
## Bag of tasks




Modern machines are multi-core and schedulers often hand out whole machines.

# The Problem

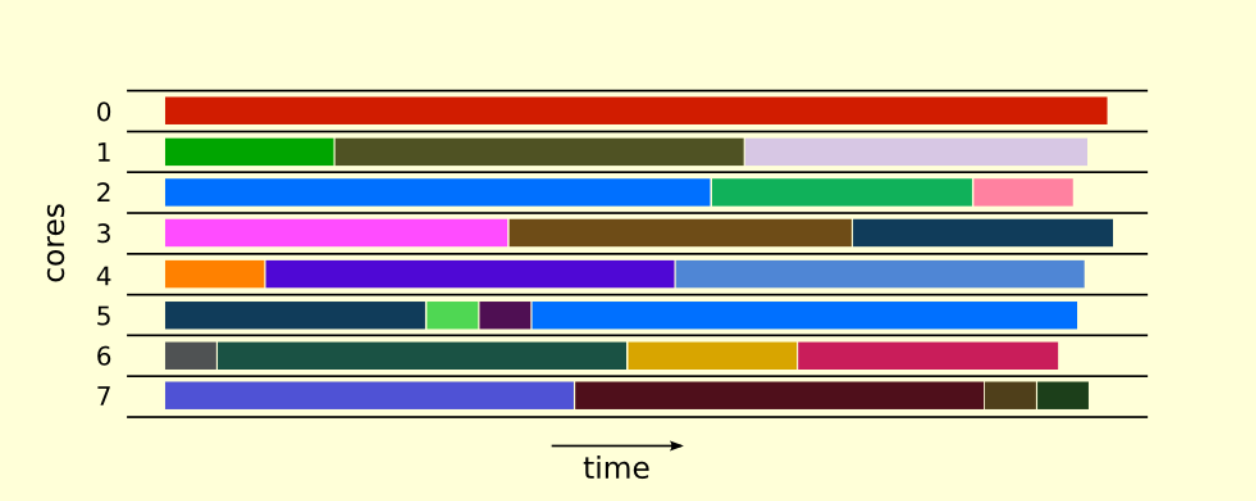
## Bag of tasks




Running one job per core does not help
if the job sizes vary significantly
(and you must now the number of cores)

# The Problem

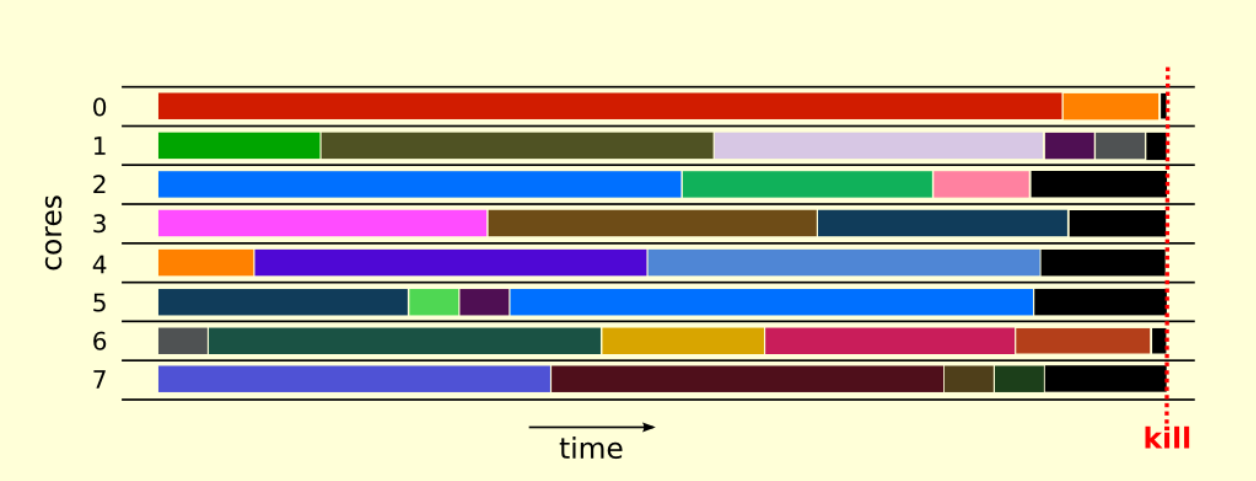
## Bag of tasks




You would like to run enough jobs to keep all cores busy for an equal amount of time (difficult to do determine schedule in advance)

# The Solution

## Bag of tasks



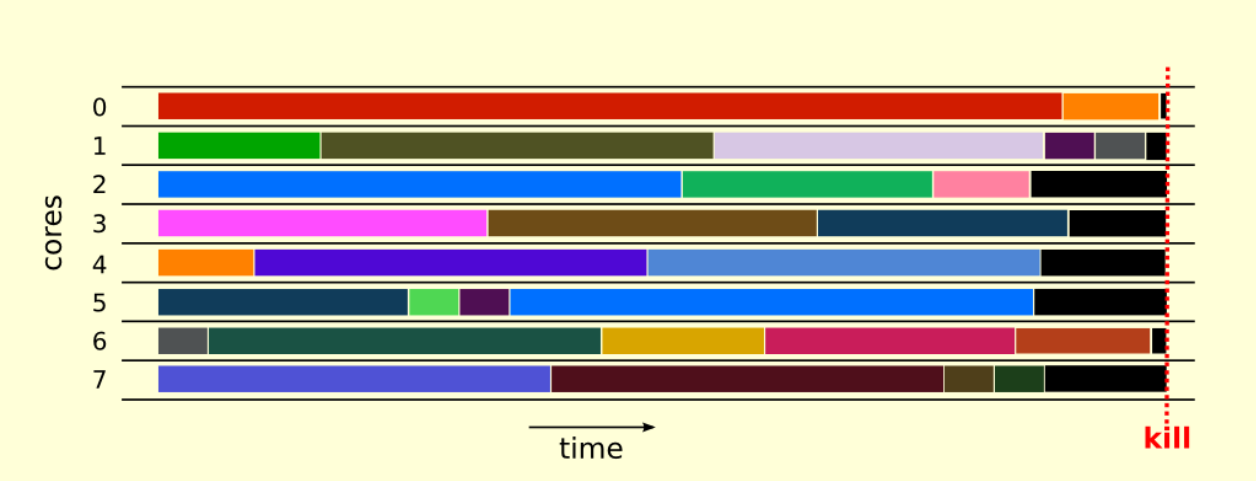

Submit application that can run a bag of tasks

Schedules jobs dynamically on the available cores

Similar to a pilot job, but no communication

# The Solution

## Bag of tasks



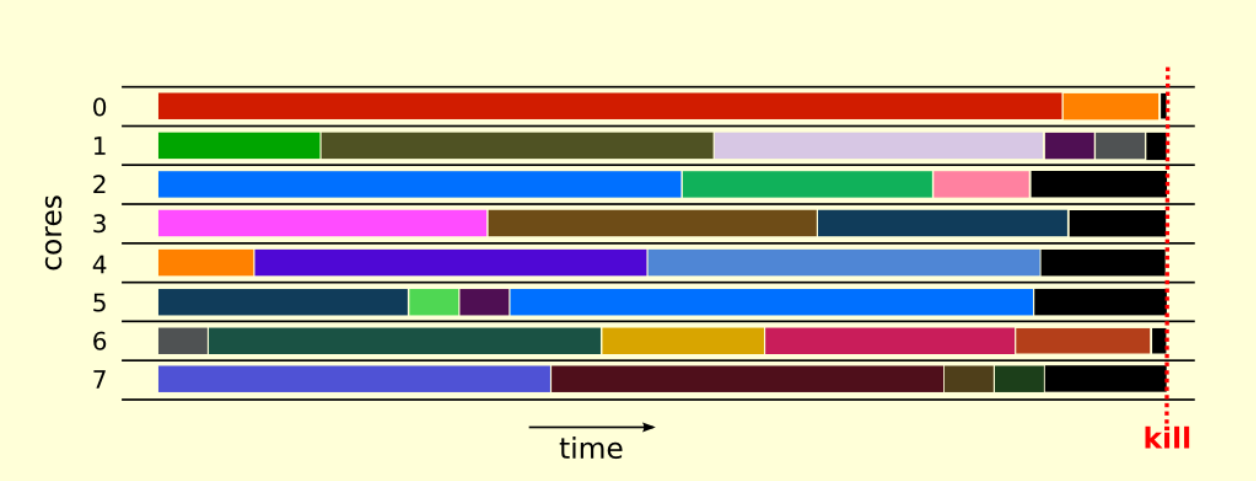

Limit the time a bag-of-tasks may run (nice for queue)

Provide more than enough jobs to keep it busy

Kills any running jobs when it runs out of time

# The Solution

## Bag of tasks




When finished we collect all results and

resubmit the unprocessed ones

in the next bag of tasks

# The code

## Bag of tasks

- See masterkey.bagoftasks package
- Main differences with previous examples:
  - Runs scripts instead of executables (more flexible)

disclaimer: this is only a very simple example!

(it only took a day to design, implement and test)

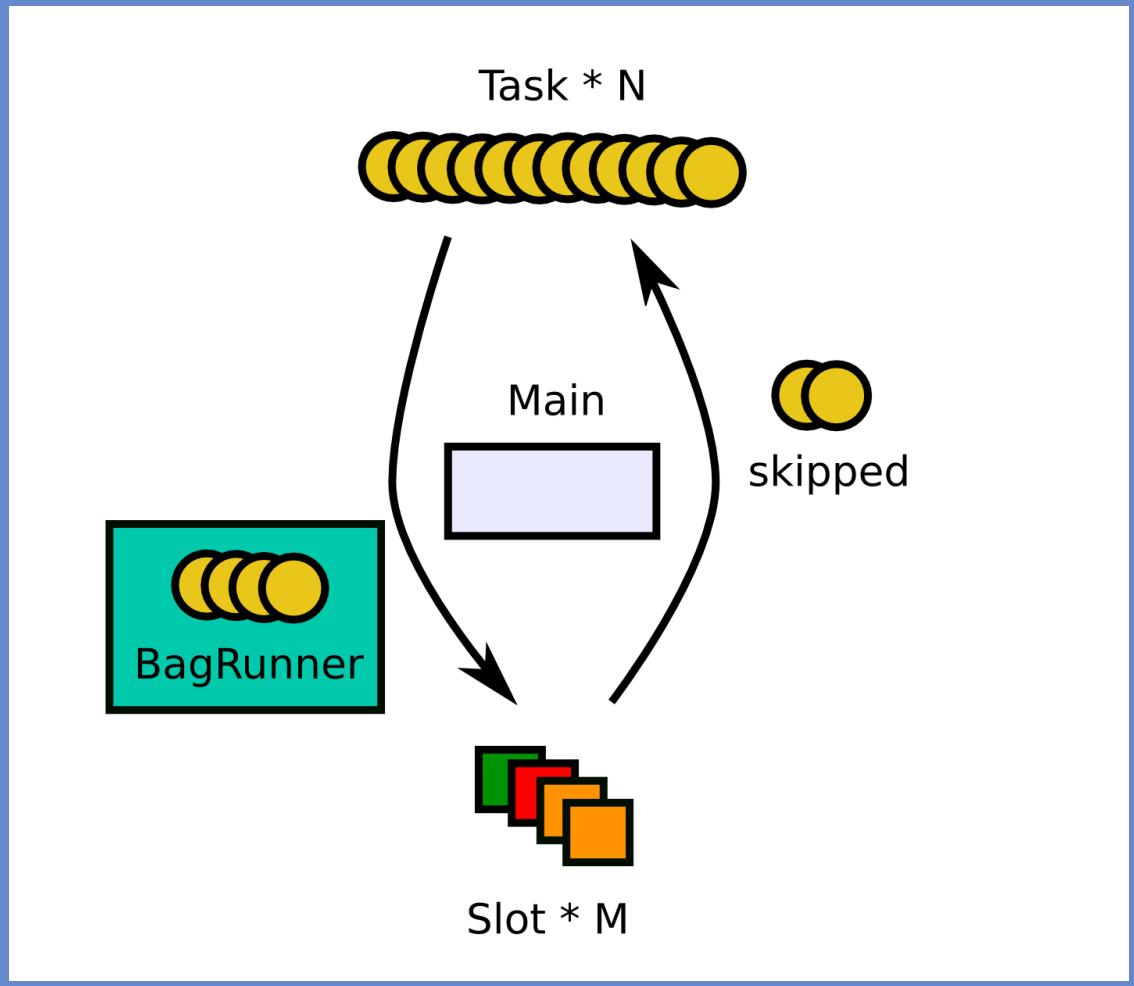
# The code

## Bag of tasks

- Code consists of 5 classes:
  - Main is in control and creates ...
  - Resources containing info about resources and a ...
  - Slot for each ‘execution slot’ on a resource and a ...
  - Tasks for each task to run and deploys a ...
  - BagOfTasksRunner to a resource to dynamically schedule a bag of tasks on the available cores.

# The code

## Bag of tasks



# The code

## Bag of tasks

```
./scripts/run masterkey.bagoftasks.Main \
  --resource sshsge://fs0.das4.cs.vu.nl /usr/bin/java 1 1 40 \
  --script ./scripts/blur.sh \
  --input ./images --output ./output
```

- The numbers after the resource are:
  - [slots] [minutes] [bagsize]
- Make sure the output directory exists!

# The code
## Bag of tasks

- After a run have a look in one of the tasks.in.* and tasks.out.* files
  - These are the in- and output of BagOfTasksRunner
- Also have a look at the stdout-*.txt files
  - These show what jobs the BagOfTasksRunner has processed an how efficient it was.