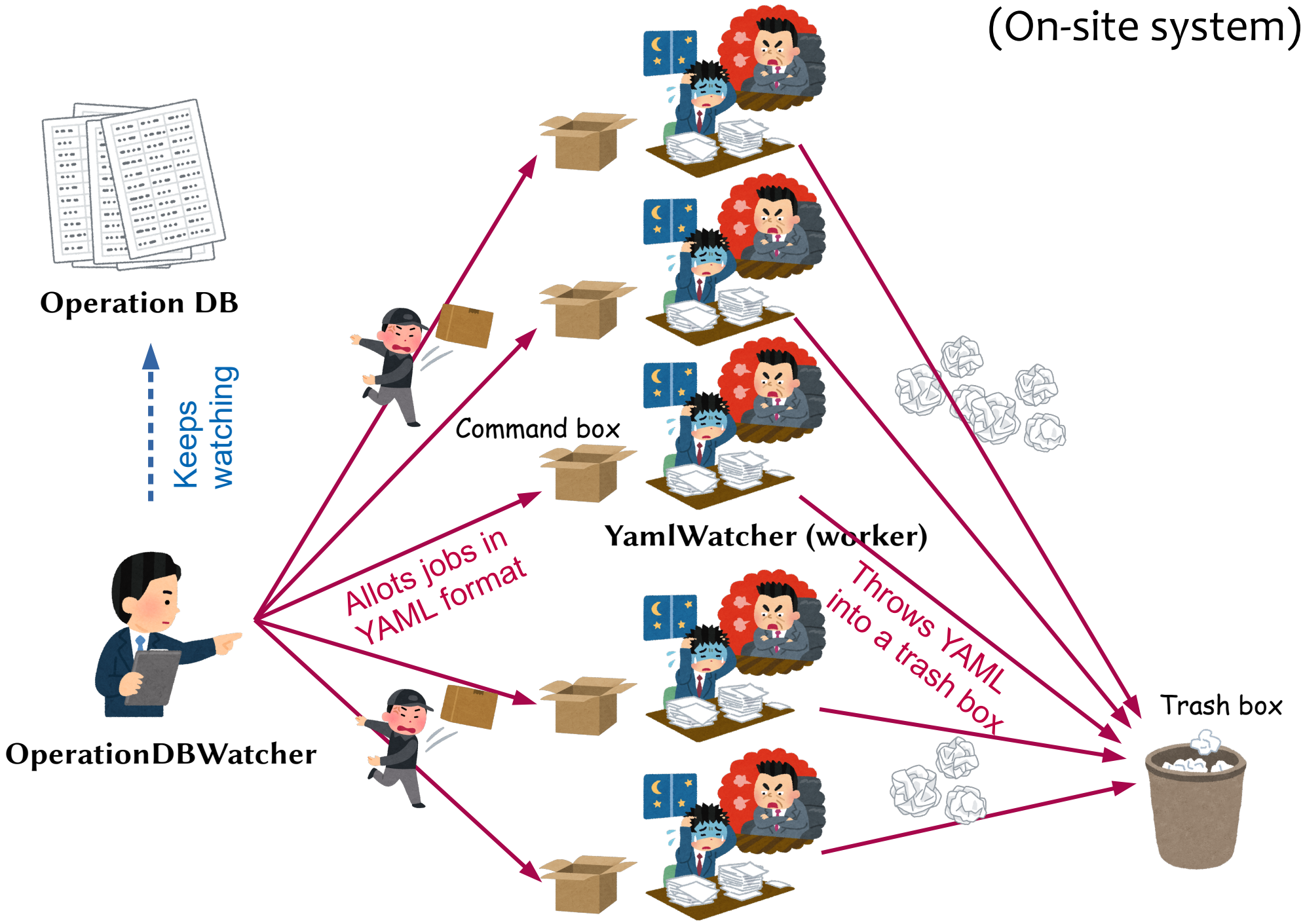
(On-site system)
Operation DB
Keeps watching
OperationDBWatcher
Allots jobs in YAML format
Command box
YamlWatcher (worker)
Throws YAML into a trash box
Trash box

Command box

- Is simply a file directory.
- Each **YamlWatcher** has one command box.

Trash box

- Is simply a file directory.
- All **YamlWatchers** share one trash box.

OperationDBWatcher

- Is a daemon process that watches the operation DB.

- Whenever a new exposure is registered in the DB,
  writes a YAML command file to one of given command boxes,
  expecting that each of these command boxes
  are watched by a **YamlWatcher**.

- Determines which command box to write the YAML command file in
  on account of how many files remain in each command boxes.

```
while True:
    exposure = waitForNextExposure(operationDB)
    cmdline = createPipelineCmdlineAsYaml(exposure)
    numJobs, workerId = min(
        (len(glob.glob(f"{dirName}{worker.id}/*.yaml")), worker.id)
        for worker in workers
        )
    with atomicCreateWriteClose(f"{dirName}{workerId}/{exposure.id}.yaml", "w") as f:
        yaml.dump(cmdline, f)
```

YamlWatcher (worker)
- Is a daemon process. Many instances.

- Watches its own command box, in which commands are posted as YAML files.
  Whenever a new command arrives, performs the command by its own thread,
  instead of delegating the task to another thread.

- After having made sure that it has done the task successfully,
  moves the YAML file to a trash box,
  which is shared by all instances of **YamlWatcher**.

- Commands can be excavated from the trash box
  in order for off-site people to reproduce the on-site products.

```
while True:
    try:
        path = waitForNextContent(f"{dirName}{myWorkerId}", "*.yaml")
        run(["onSiteDriver.py", path])
        shutil.move(path, trashBox)
    except Exception:
        sys.excepthook(...)
```

onSiteDriver.py

- Performs on-site data processing.
  Designed for on-site analysis, but can be executed anytime anywhere.

- Has the common command line interface:
  onSiteDriver.py ROOT --calib=CALIB --rerun=RERUN --id=ID

- Can also eat a single YAML file that represents a common command line.

  onSiteDriver.py args.yaml

- Starts with creating calibration files if they are not found.
  There must be optional arguments like `--bias-from=ID` , `--dark-from=ID` , etc.,
  which are ignored if suitable calibration files are found.

Why should there be both **YamlWatcher** and `onSiteDriver.py`?

- onSiteDriver.py should be able to be run manually just like other executables in the LSST Stack.
  It should not have both this manual operation mode and a daemon-type operation mode.

- **YamlWatcher** is devoted to feeding a YAML file to `onSiteDriver.py`
  and moving the YAML file from a command box to the `trash box`.

Why should there be both **OperationDBWatcher** and **YamlWatcher**?

- If onSiteDriver.py fails, operators modify the YAML file in the command box.
  YamlWatcher, noticing file timestamp change, starts retrying.
  These two entities should be separated in order for this manual error handling.