Task Distribution Framework

Benjamin Schiller, P2P Networks, TU Darmstadt

Research Meeting / 04.07.2012

Motivation

- Distributed task processing
  - Makes large-scale computations scalable
  - Facilitates load distribution between multiple machines
  - Especially interesting for distributed data collection

- Applications
  - Web crawling / data collection
  - Distributed computations
  - Collaborative problem solving

Requirements

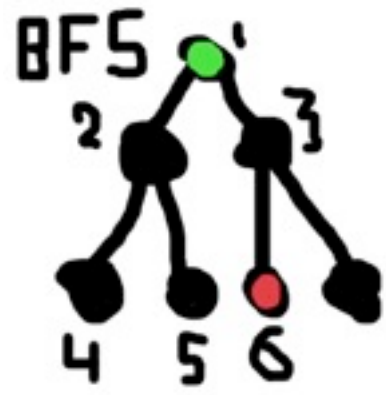
- Distribution of tasks from a centralized server to clients
  - Clients execute task and return result(s)
- Task management
  - Tasks have pre-defined lifecycle
  - Re-queueing in case of failures
- Task execution independent of API / programming language
  - Automated download of required software
- Execution of different task types

Assumptions

- Tasks can be executed separately
- No client-to-client communication required
- No sub-tasks (queue, not tree)
- Running on *nix systems
- Single server instance (SPoF)
- Simple HW/SW restrictions handled using namespaces

Example

- BFS crawl of a social network
  - Task: crawl profile of user A
  - Start: server creates task for initial user I
  - Client: get task, fetch profile, return to server
  - Server: add new tasks based on profile's edges in result

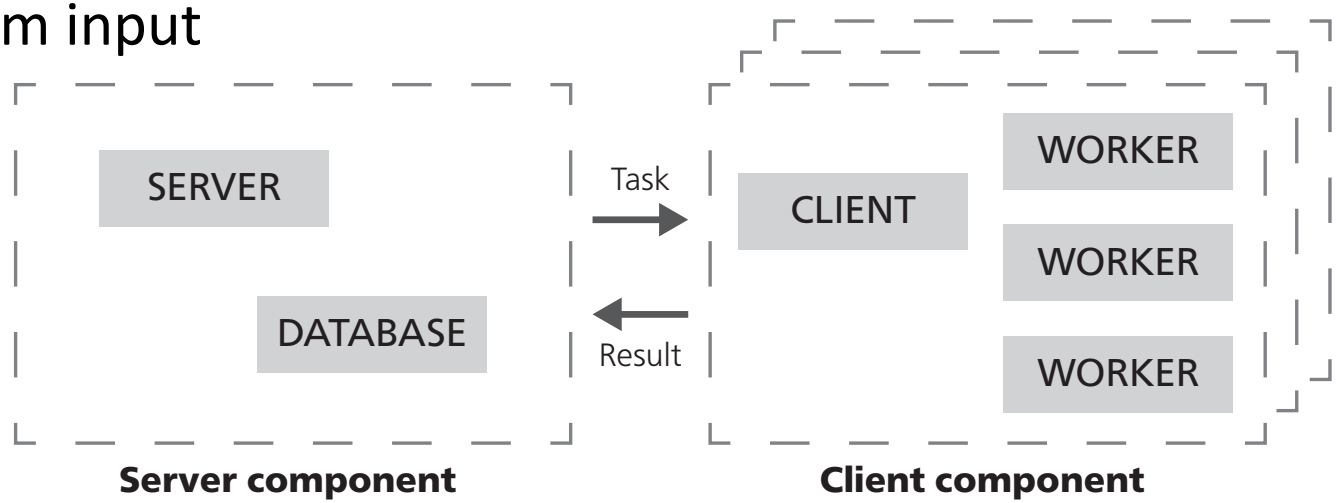


- Compute the sum of 1, 2, ... n
  - Task: add a_1, a_2, ... a_k
  - Start: server creates tasks, e.g., $T_1 = (1,2,3)$, $T_2 = (4,5,6)$, ...
  - Client: get task, compute sum(a_1, a_2, ... a_k)
  - Server: after receiving k results, create new task T = (r_1, r_2,... r_k)

Terminology

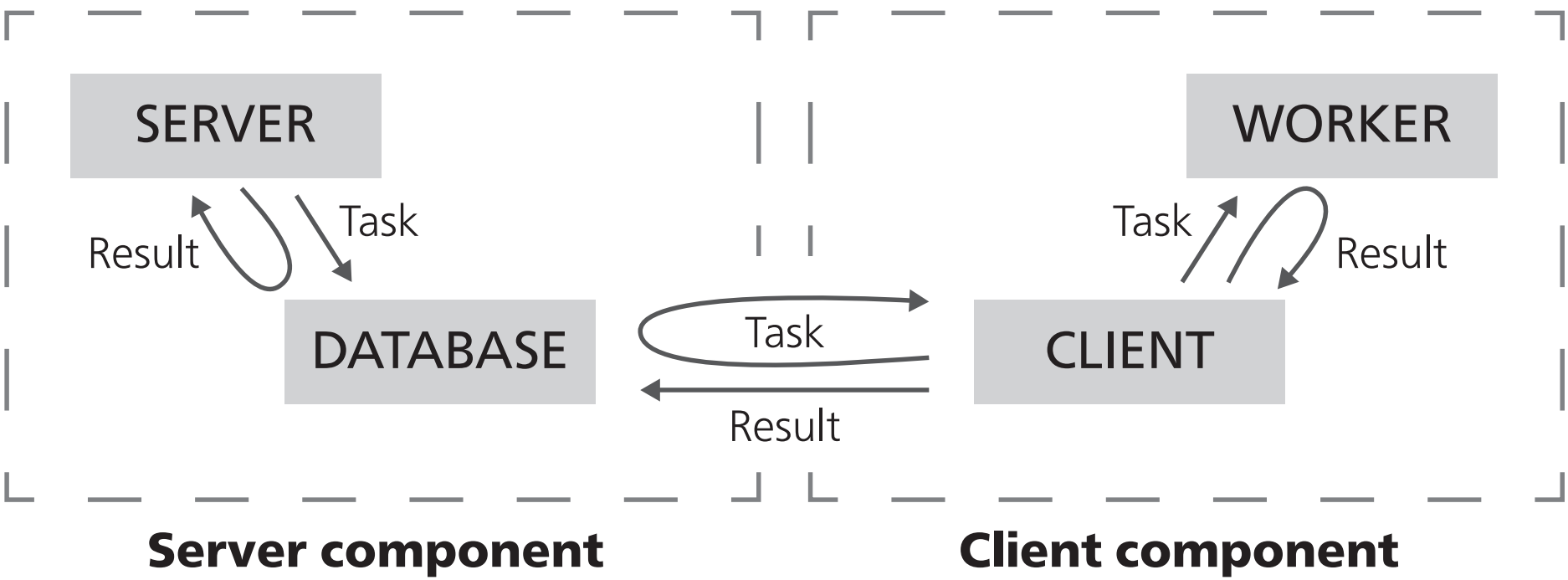
- Server
  - Creates and manages tasks

- Worker
  - Executes a given task
  - Generates result from input
- Client
  - Retrieves task
  - Fetch required worker
  - Return results to server



Communication Paradigm

	Task	Result	Server	Client
1	Push	Push	active	active
2	Push	Pull	active	passive
3	Pull	Push	passive	active
4	Pull	Pull	active	active



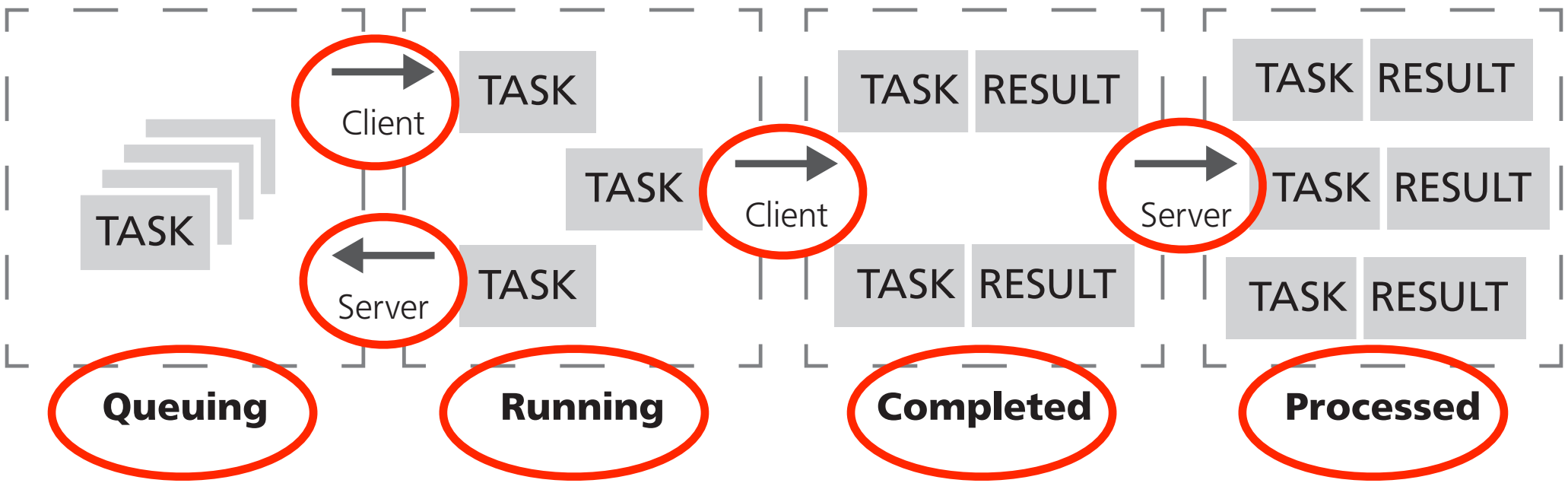
Task Attributes set by Server

- **worker** - *https://dl.dropbox.com/u/2187880/TUD/gplus-client-v-2.zip*
- **input** - *123 456 789 [file]*
- **session** - *2012-07-04 [String]*
- **runAfter** - *2012-07-04 00:00:00 [DateTime]*
- **runBefore** - *2012-07-04 23:59:59 [DateTime]*
- **timeout** - *10.000 [ms]*
- **waitAfterSuccess** - *1.000 [ms]*
- **waitAfterSetupError** - *100.000 [ms]*
- **waitAfterRunError** - *5.000 [ms]*

Task Attributes set by Client

- **output** - *profile(123), profile(789) [file]*
- **log** - *fetched 123 fetched 789 [file]*
- **error** - *error fetching 789 [file]*
- **client** - *lab2 [String]*
- **started** - *2012-07-04 14:27:25 [DateTime]*
- **finished** - *2012-07-04 14:27:31 [DateTime]*

Task Lifecycle



Task Lifecycle

Re-queuing

```
for each task in namespace.running
    if task is timed out
        remove task.index from namespace.running
        clear task.client
        clear task.started
        lpush task.index to namespace.queuing
```

Delete expired tasks

```
for each task in namespace.queuing
    if task is expired
        remove task.index from namespace.queuing
        delete task
for each task in namespace.running
    if task is expired
        remove task.index from namespace.running
        delete task
```

Server API

- **getNamespaces**(), **addNamespace**(ns), **deleteNamespace**(ns)
- **deleteAllTasks**(ns)
- **increaseIndexForNamespace**(ns)
- **addTask**(ns, task), **deleteTask**(ns, index), **getTask**(ns, index)
- {**get delete count**}{**Queueing Running Completed Processed**}Tasks(ns)
- **processTask**(ns, index)
- **deleteAllExpiredTasks**(ns), requeue(ns)
- **exportProcessedTasks**(ns, dir, input, output, log, error, information)
- **close**()

Configuration

db.properties

redis.host = localhost
redis.port = 6379
redis.index = 14
*redis.auth = ***********

client.properties

client.id = "client-1"

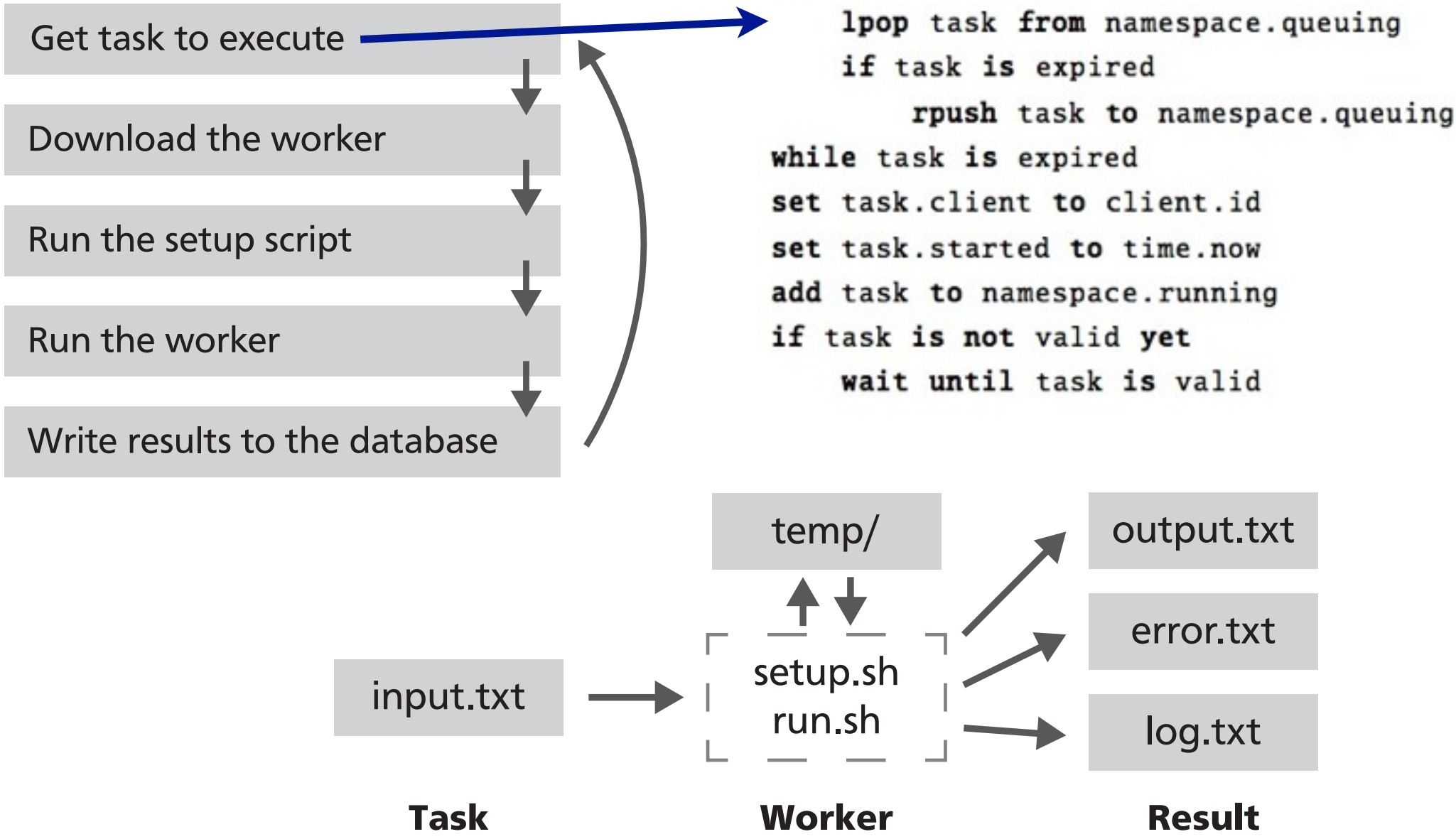
wait.queue.empty = 2000
wait.queue.expired = 10000
wait.success = 0
wait.error.setup = 0
wait.error.run = 0

redis.host = localhost
redis.port = 6379
redis.index = 14
*redis.auth = ***********

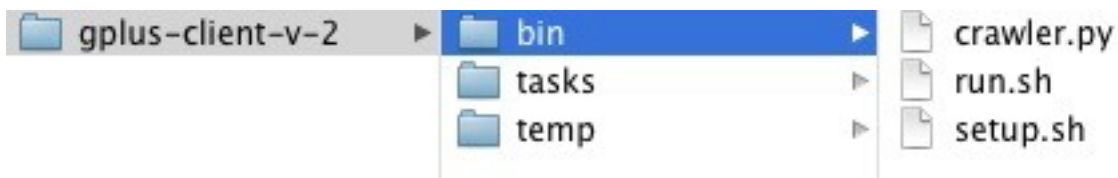
worker.dir = ./runner

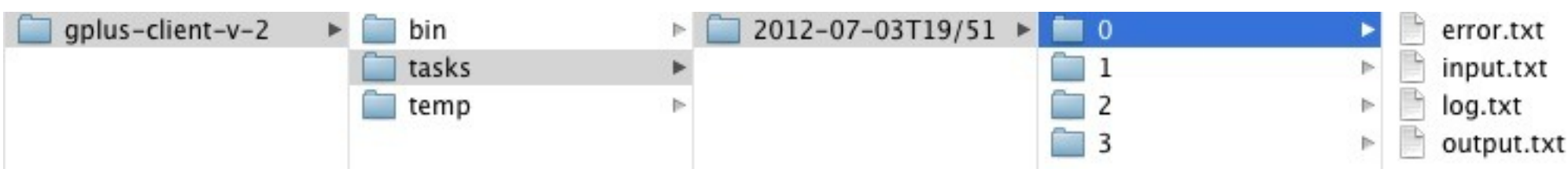
namespaces =

Client Workflow

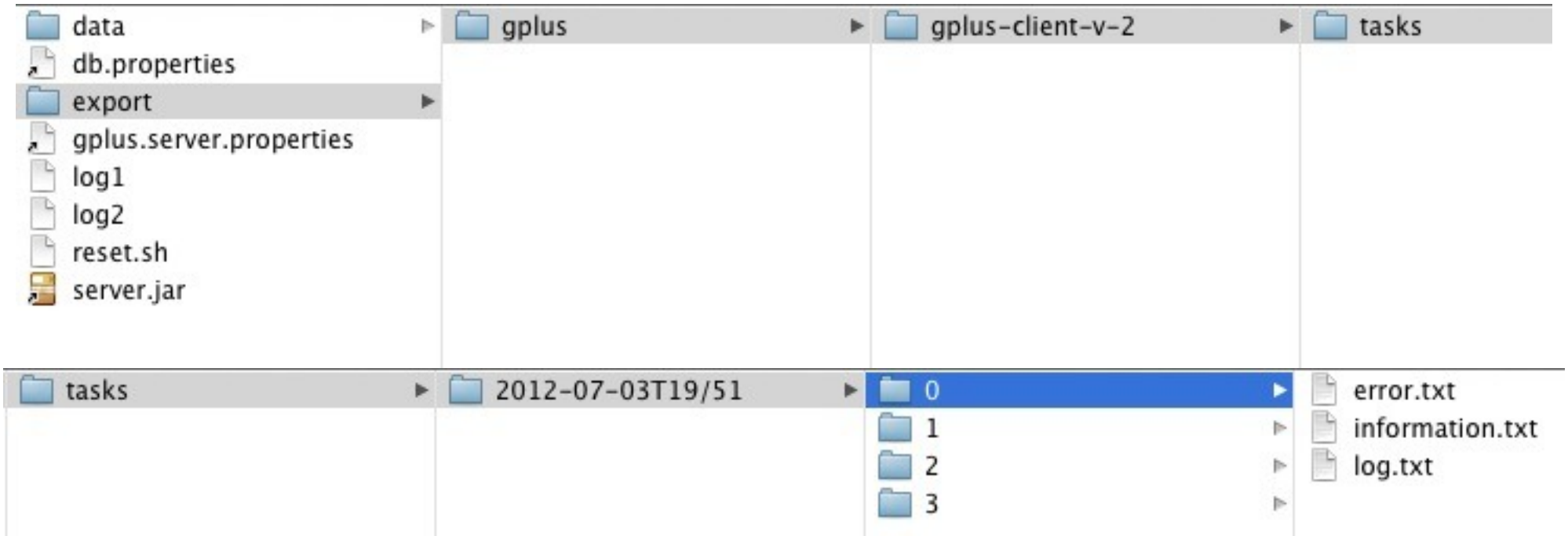


Client Datastructures





Server Datastructures



worker: https://dl.dropbox.com/u/2187880/TUD/gplus-client-v-2.zip
client: "client-1"
started: 2012-07-03T19:51:32.000+02:00
finished: 2012-07-03T19:51:36.000+02:00
runAfter: 2012-07-03T19:51:00.000+02:00
runBefore: 2012-07-03T19:56:00.000+02:00
timeout: 100000
waitAfterSuccess: 100
waitAfterSetupError: 100000
waitAfterRunError: 1000
index: 0
namespace: gplus
session: 2012-07-03T19:51

Implementation

- Database server
  - redis version 2.4.4

- Java Server API
  - Java version 1.6, Jedis version 2.0.0

- Python Server API
  - Python version 2.7.1, redis-py version 2.4.13

- TDF client
  - Java version 1.6

Related Work (1/2)

- BOINC / SETI@home
  - Communication over RPCs, check client restrictions
  - C(++) API, Java wrapper exists
  - **Push / Push**
- Hadoop Distributed File System (HDFS)
  - Stores large files across multiple machines (replication)
  - JobTracker schedules map/reduce jobs (location-aware)
  - JVM languages only
  - **Push / Push** (storage in HDFS)
- DisPyTE
  - Modular task distribution framework, focus on load balancing
  - Python binding required
  - **Push / Push** (callback)

Related Work (2/2)

- Banyan
  - Framework for solving tree-structured search problems
  - Supports Scala & Java
  - **Push / Push** (stored in shared memory)
- FALKON
  - High performance task execution framework
  - Up to 50k executor nodes
  - **Hybrid (Push/Pull) / Push**
- Celery
  - Distributed asynchronous queue for tasks as Python modules
  - **Push / Push** (written to result store)

Summary & Outlook

- TDF - Task Distribution Framework
  - Light-weight distribution of arbitrary tasks to client
  - Pull of tasks, push of results
  - Simple I/O-based interface between TDF and worker
  - Automated update of worker software
- Applications so far
  - Sum calculation, stock crawling (first examples)
  - Freenet churn analysis
  - G+ crawler (v4)
- Outlook
  - Download of worker software from git/svn repositories
  - Evaluation of task statistics
  - Logo ;-)