WHITE
RABBIT

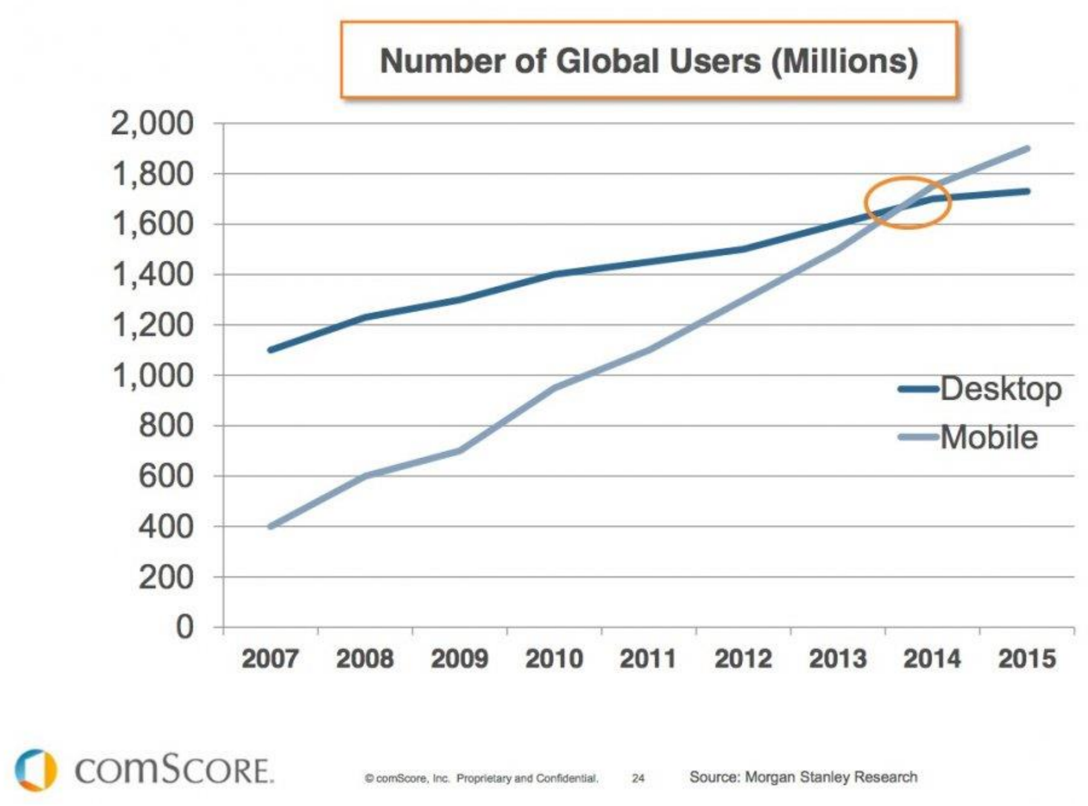
A Volunteered Computing Platform

An opportunistic use of CPU cycles from mobile devices

KMUTT
King Mongkut's
University *of*
Technology
Thonburi

Smartphone today

- Smartphones and tablets are becoming increasingly powerful and rising quickly in popularity.

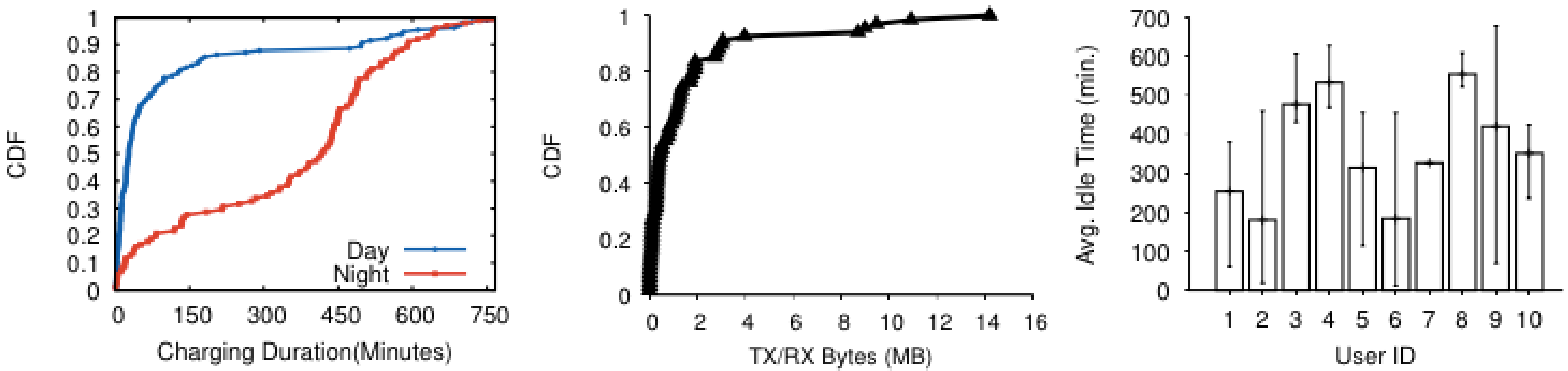


Computing device sales comparison

Smartphone charging behaviors

- **A study on the availability of task execution periods (presented in the CWC project)**
  - Identify and attempt to utilize idle periods of smartphones
  - Profile the charging behaviors of users through an Android App
    - 3 states —> plugged, unplugged and shutdown
    - Tracks total bytes transmitted and received over WIFI and cellular network

Smartphone charging behaviors (2)

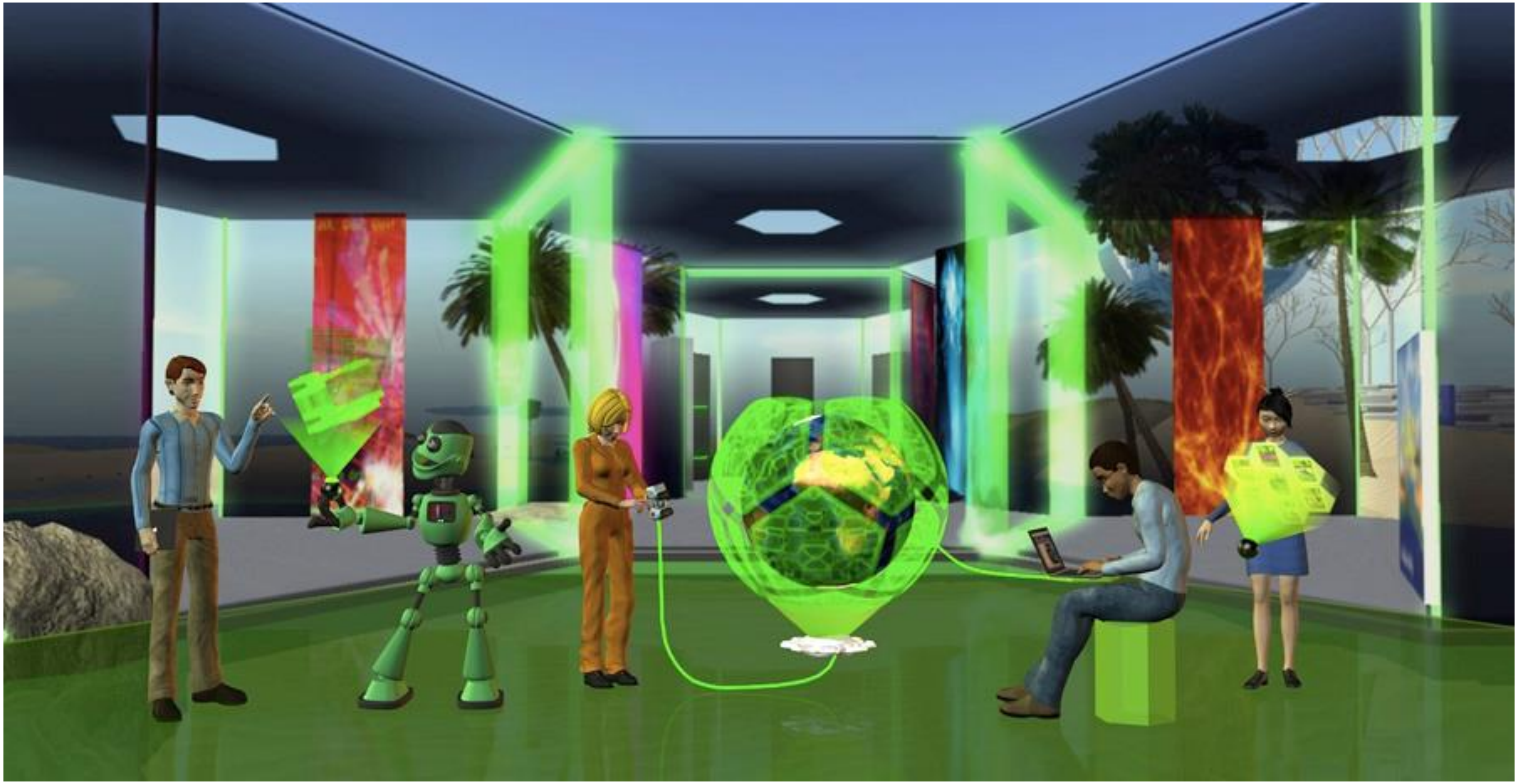


(a) Charging Duration (b) Charging Network Activity (c) Average Idle Duration

Charging interval: **day—> 30 mins** and **night —> 7 hours**
User is unlikely to be actively using the phone at night, less than 2 MB
Users have at least 3 hours of idle charging at night

Volunteered Computing

- *Donation of CPU cycles to help solving scientific problems*



www.volunteer-computing.org

Key Idea of White Rabbit

Volunteered computing power for technological advancement in Sciences

- Make use of idle computing resources.
- Promote the importance of Sciences.
- Create a new educational channel for sciences.


White Rabbit will bring ALICE home to you

White Rabbit: Aims & Objectives

- To promote the ALICE Experiment to the communities (in Asia and Europe)

- To promote sciences to young generations

- To build a light weight mobile volunteered computing framework

- To aggregate computing power of smartphones and exploit the wasted cycles of those devices while we sleep

White Rabbit: The Plan for 2015-16

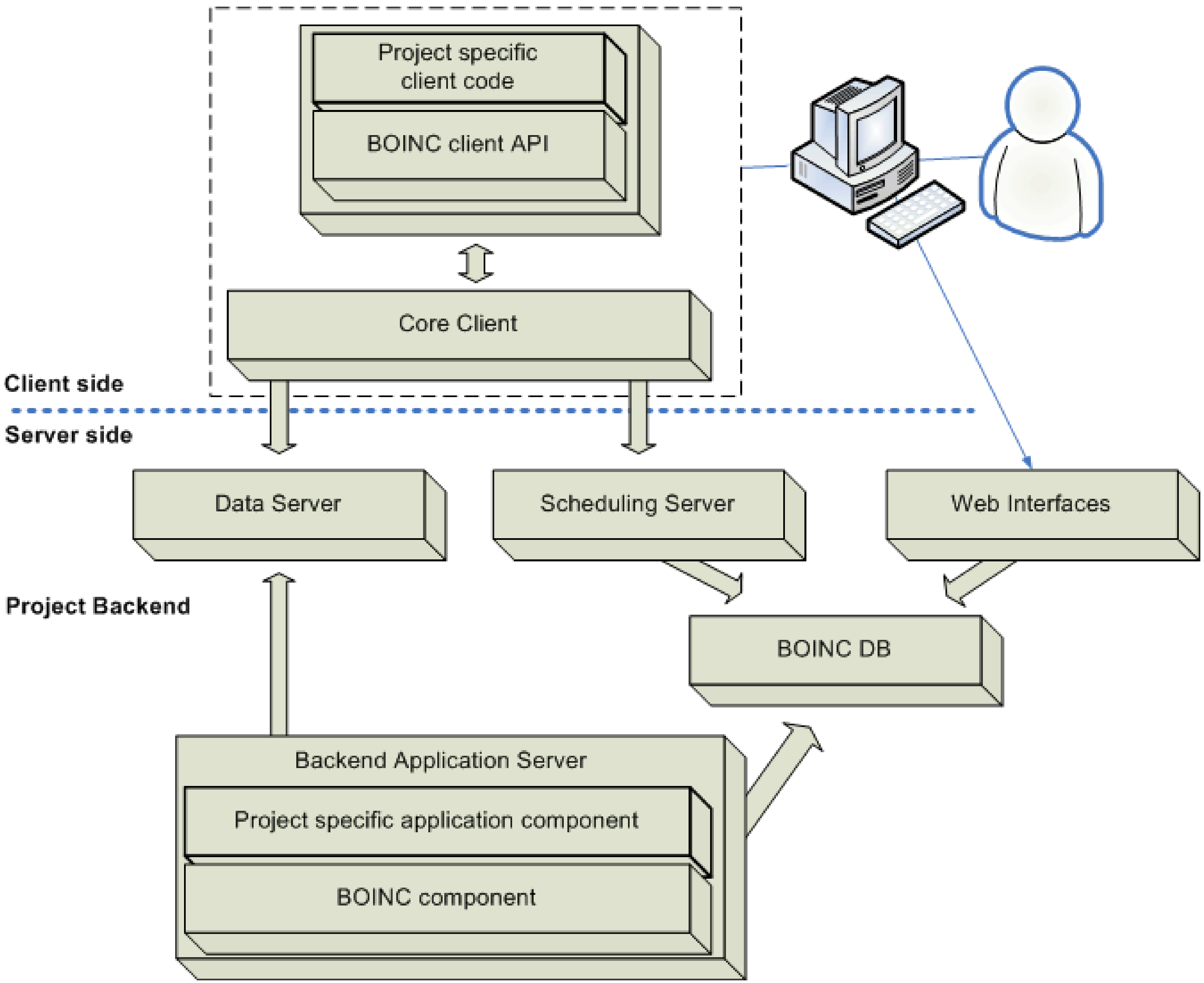
- Study, design and deploy a mobile volunteered computing platform (Most likely based on the BOINC framework).
- Port a few of applications in the ALICE experiment (such as "TOF detector calibration") onto White Rabbit
  - Validate the protocol
  - Evaluate the performance
- Design new services: "rewarding scheme", "social network Enabled", and "education delivery"
- Deliverables
  - White Rabbit (Implementation and Deployment)
  - New services as add-on modules
  - 1-2 Publications

Notes on Related Technologies

The Berkeley Open Infrastructure for Network Computing

Use the idle time on your computer to cure diseases, study global warming, discover pulsars, and do many other types of scientific research.

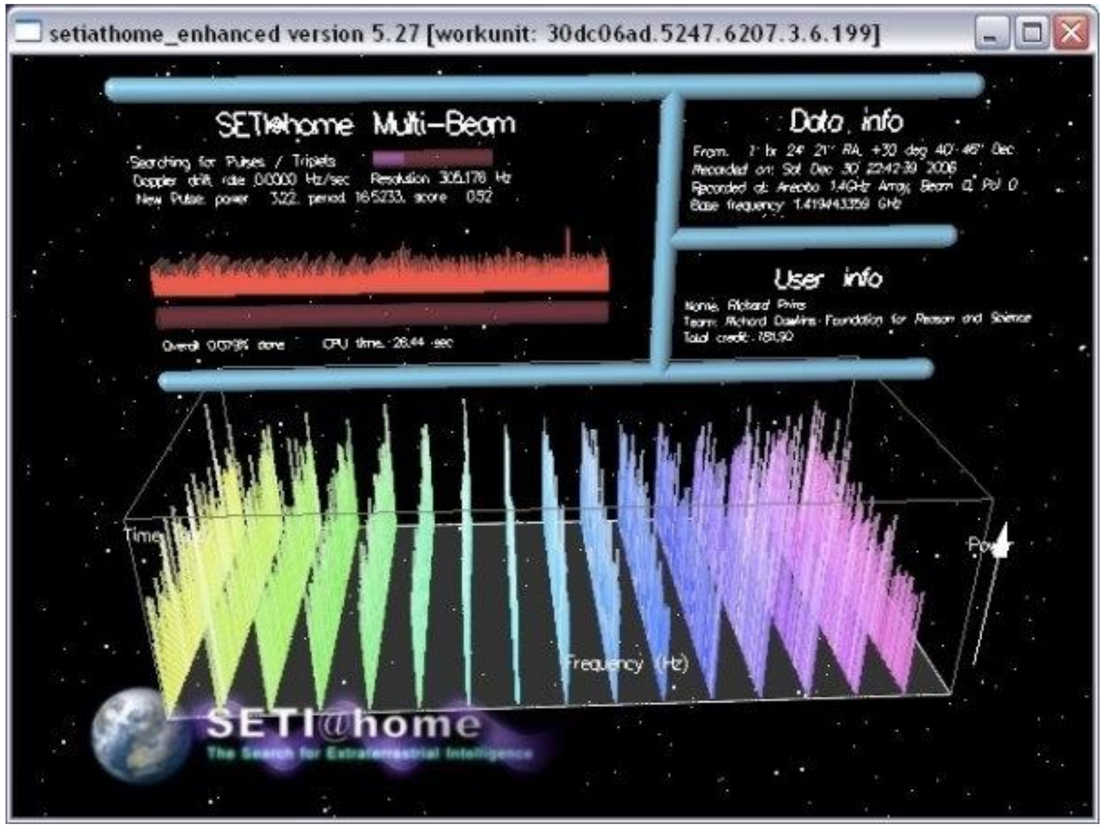
BOINC


Project specific
client code
BOINC client API
Core Client
Client side
Server side
Data Server
Scheduling Server
Web Interfaces
Project Backend
BOINC DB
Backend Application Server
Project specific application component
BOINC component

BOINC‘s Adoptions

SETI@Home
- 3 million participants
- 600 TFLOPS

Folding@home
- 300,000 contributors
- 5 PFLOPS sustained



A research project that uses volunteered computing to run simulations of the ATLAS experiment

- **Hardware**
  - A reasonably powerful modern 64-bit computer with at least 4GB of memory is required.
- **Software**
  - VirtualBox ~500MB
  - BOINC Client
- Each work unit downloads a small set of input data and runs for approximately 1 to 2 hours depending on the computer's processor speed.

Virtual LHC@home

- The Virtual LHC@home project (formerly known as ***Test4Theory***) allows users to participate in running simulations of high-energy particle physics using their home computers.
- The results are submitted to a database which is used as a common resource by both experimental and theoretical scientists working on the Large Hadron Collider at CERN.

Limitations of BOINC

- The BOINC server can only be executed on GNU/Linux-based operating systems.
- The platform is relatively heavy with lots of embedded modules.
- Researchers creating BOINC projects must learn the BOINC programming API and be proficient in
  - Linux system administration
  - MySQL administration
  - The Extensible Markup Language (XML), and C++.
- Limited documentation and very few tools to facilitate the creation of new projects, resulting in a long, manual process.

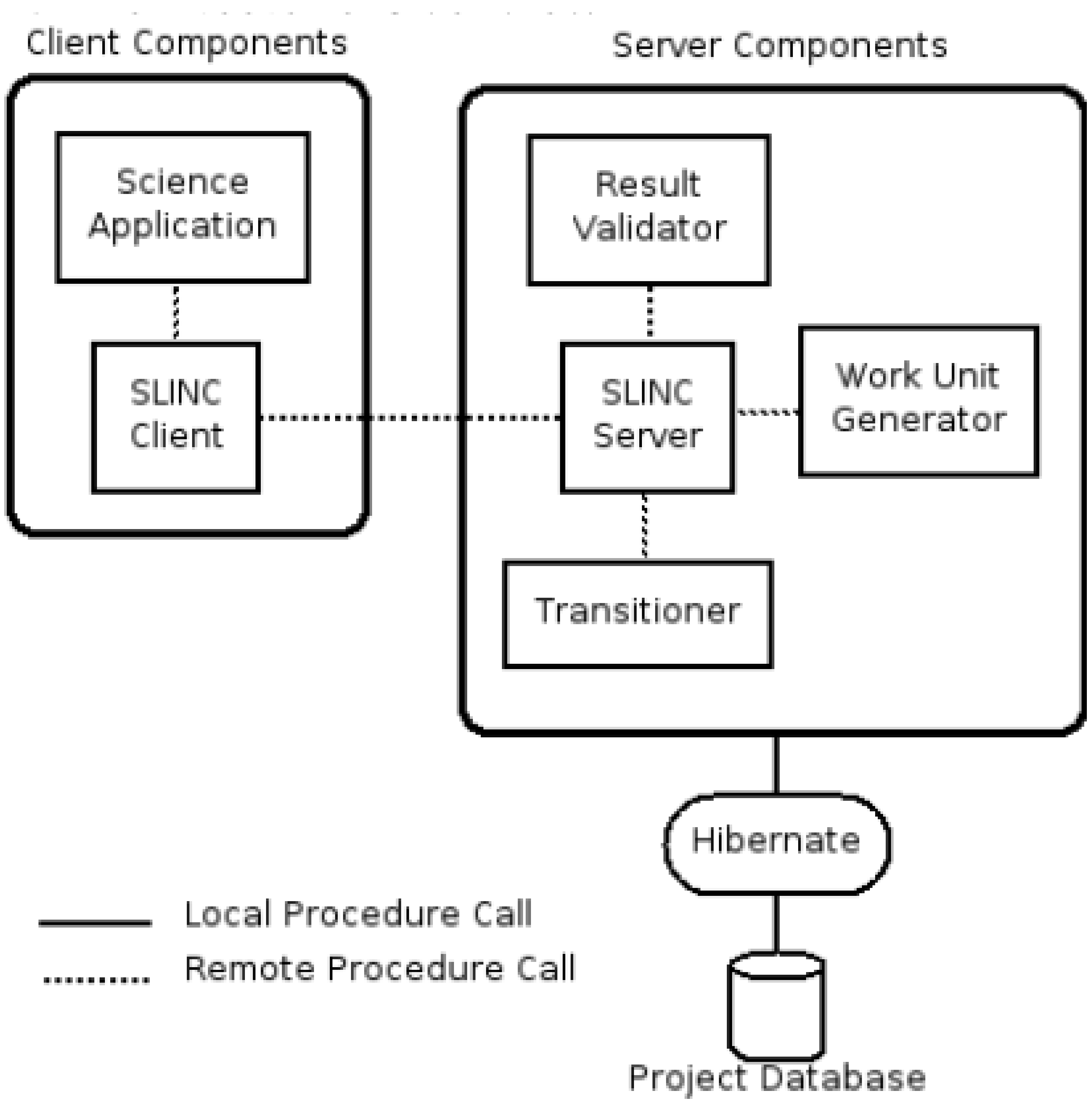
SLINC

Simple Light-weight Infrastructure for Network Computing

The existing volunteer computing frameworks are too complex, limiting, and difficult to use for scientists.

- **Goal:** to create a new framework that simplifies the process of creating volunteer computing projects. The framework should be scalable with modular and object-oriented design.
- **Server**: partitioning input data into work units, distributing work units to clients, and processing and validating results for each work unit.
- **Client**: request work units from the project server, compute the result for each work unit, and return the result to the server.
- Each module can be located on different physical computers and all components can communicate via XML-RPC

SLINC: Architecture



Other Related Works

- **Ibis** : an open source Java based high performance distributed computing platform with a version on Android.
- **AVRF** : Android Volunteered Resource Framework, designed to allow Android phones to act as volunteer workers for distributed computing tasks.
- **Hyrax** : A cloud computing platform on mobile devices using the Mapreduce concept.

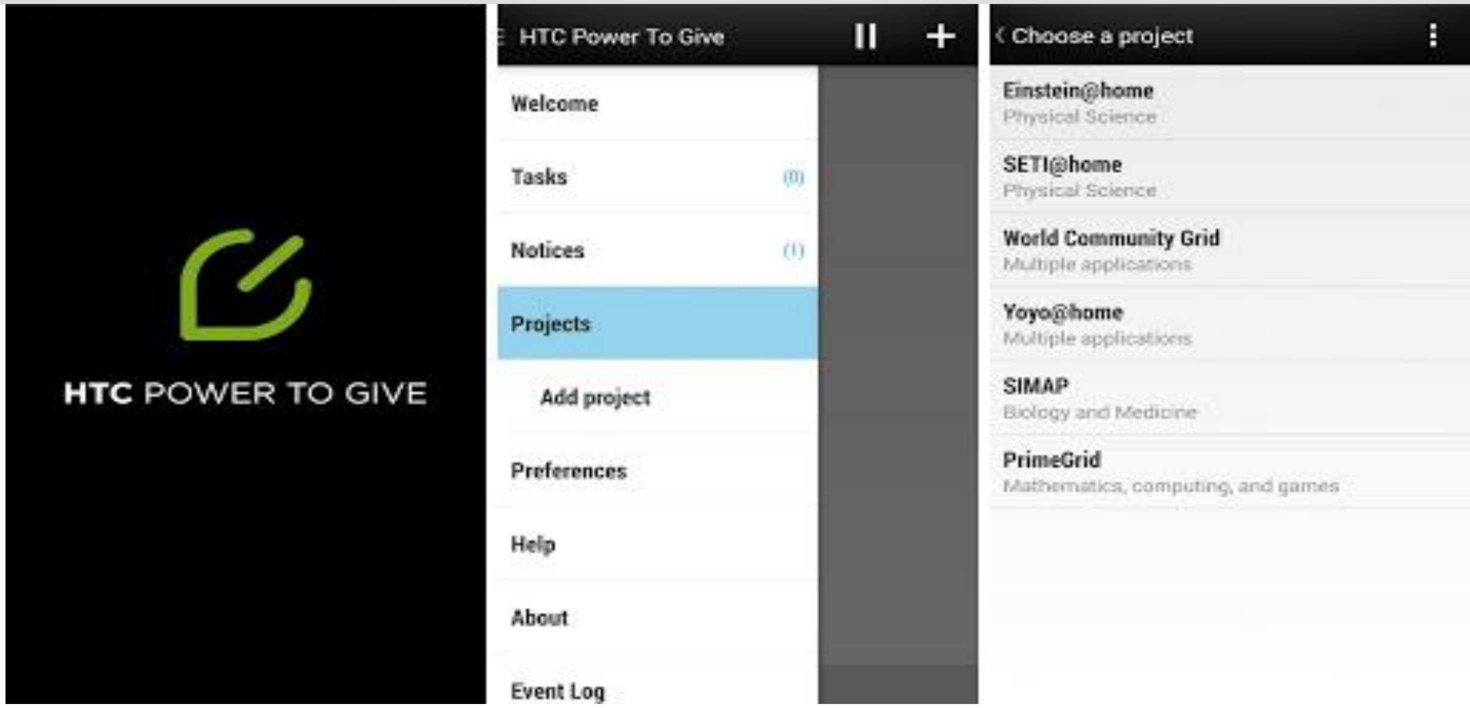
Why BOINC?

- It’s the only platform with multiple actual usages in mega-science projects.
- It’s a mature platform with relatively strong supporting communities.
- So…...this is an easy decision!

BOINCOID

A Project aims at creating a volunteered computing platform on ARM-based mobile devices.

HTC Power to Give



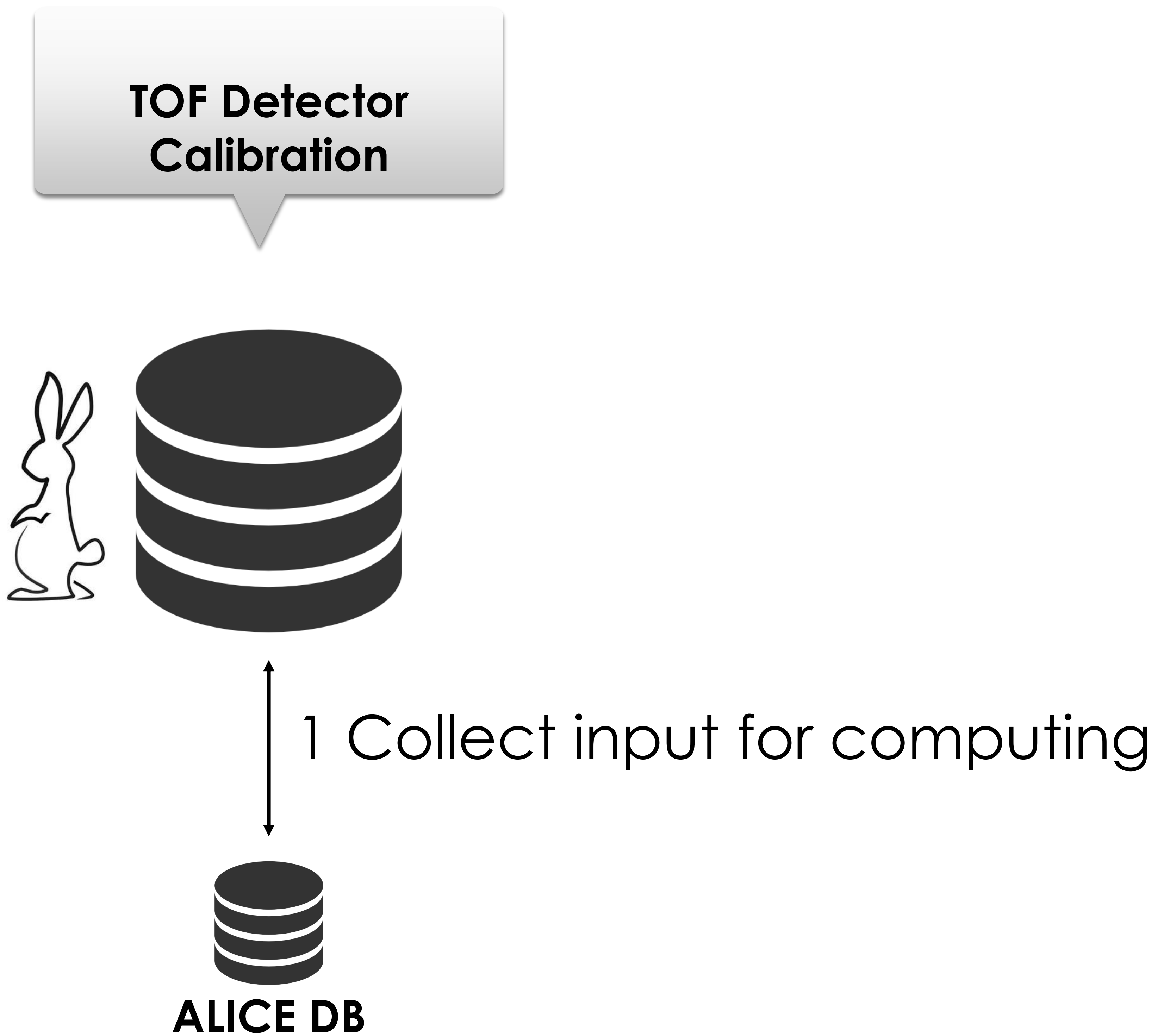
Samsung Power Sleep

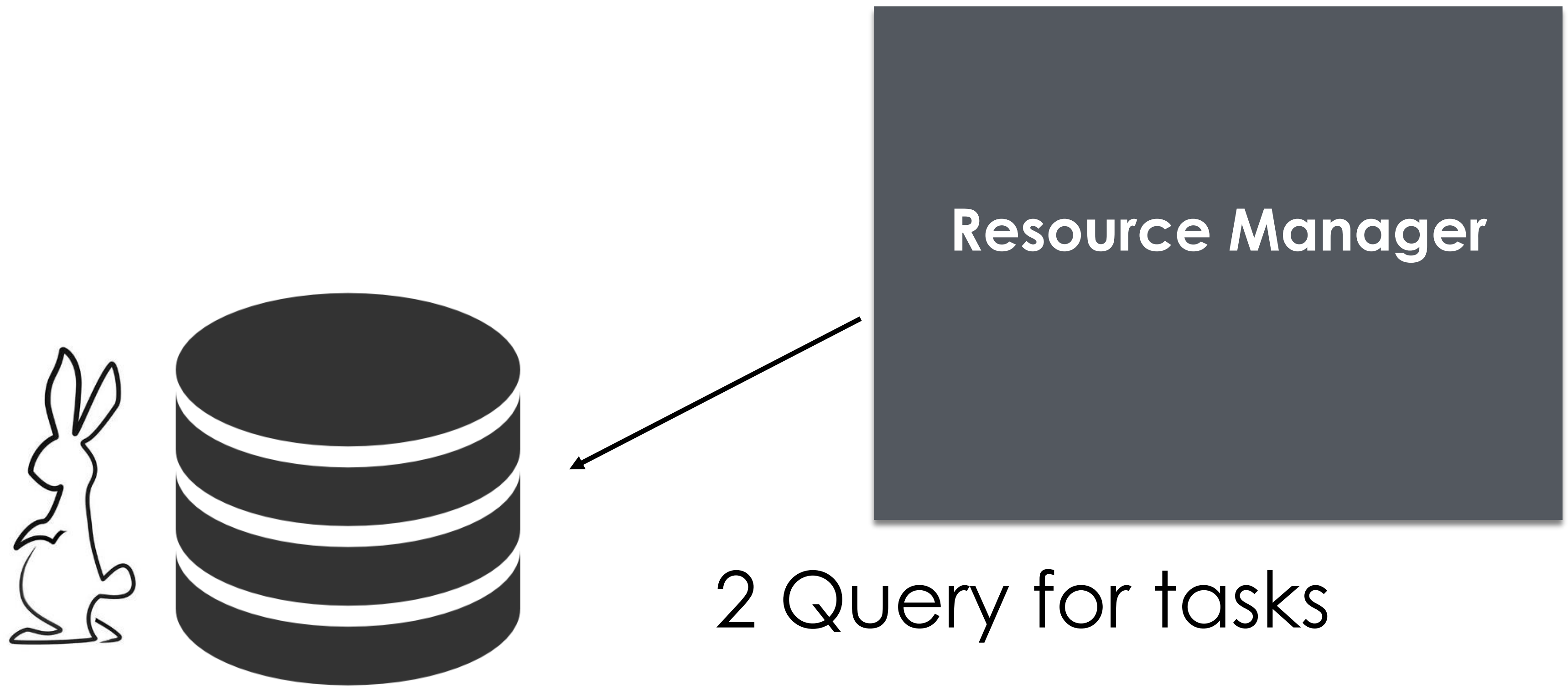
Volunteered Computing on iPhone

- Technical and legal barriers
  (It's possible that in the future these issues can be overcome.)
- Multitasking issues
- Ability to control hardware resources on devices

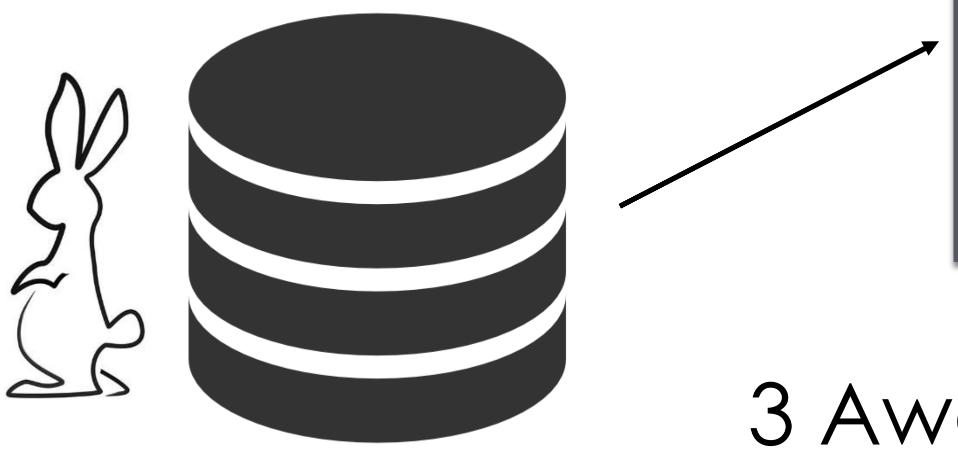
The 10 steps of

WHITE
RABBIT


TOF Detector
Calibration
1 Collect input for computing
ALICE DB



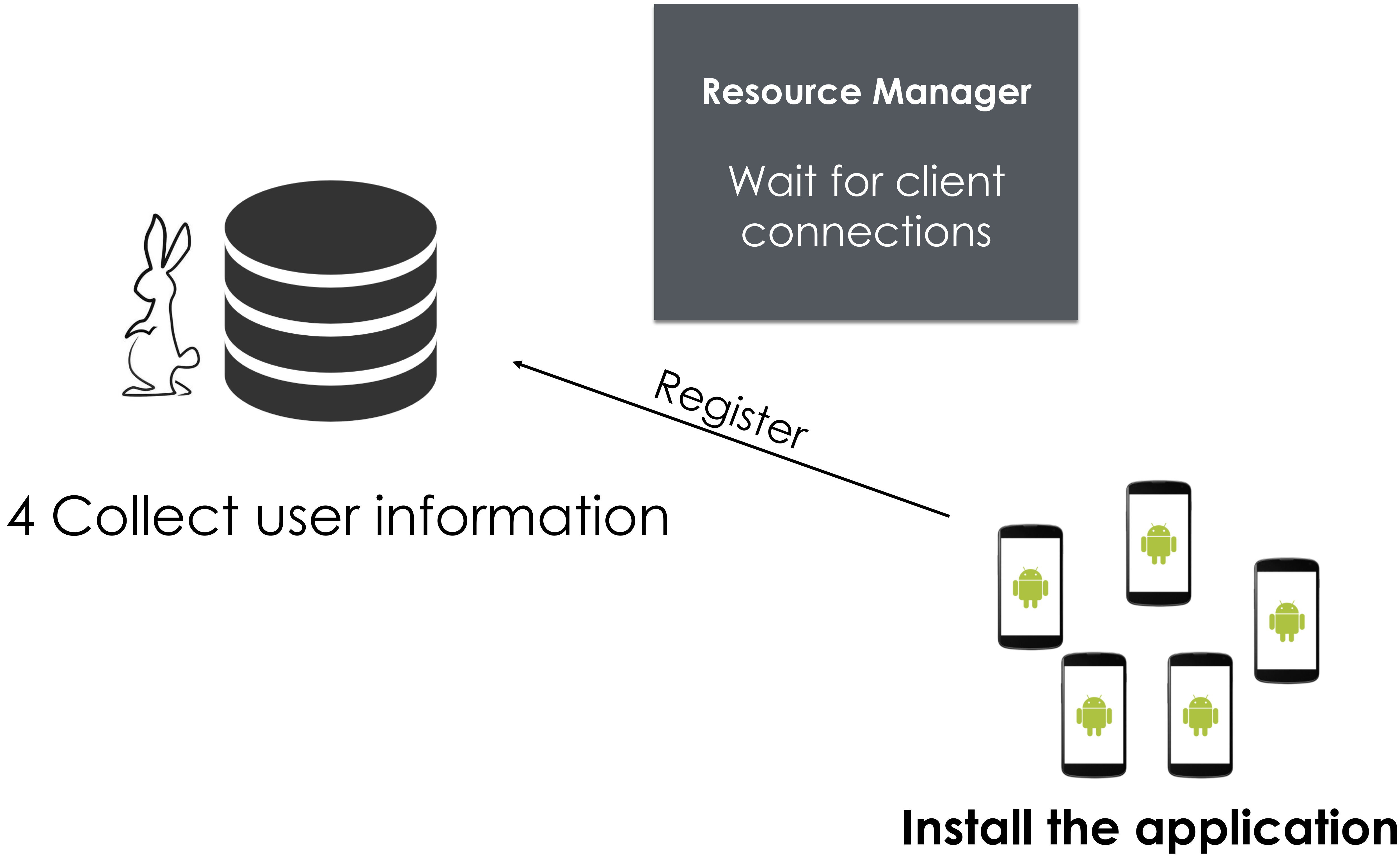
2 Query for tasks

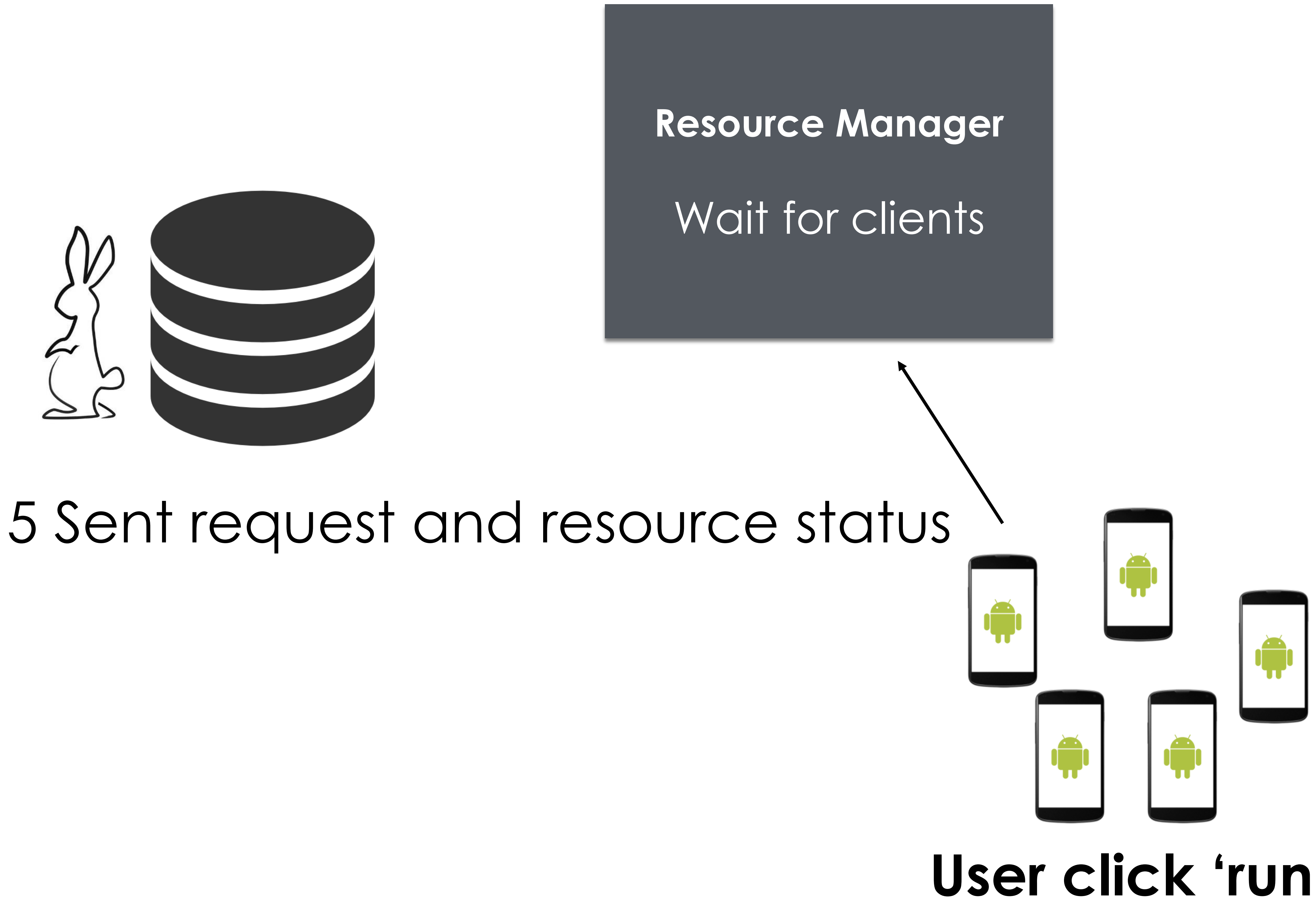


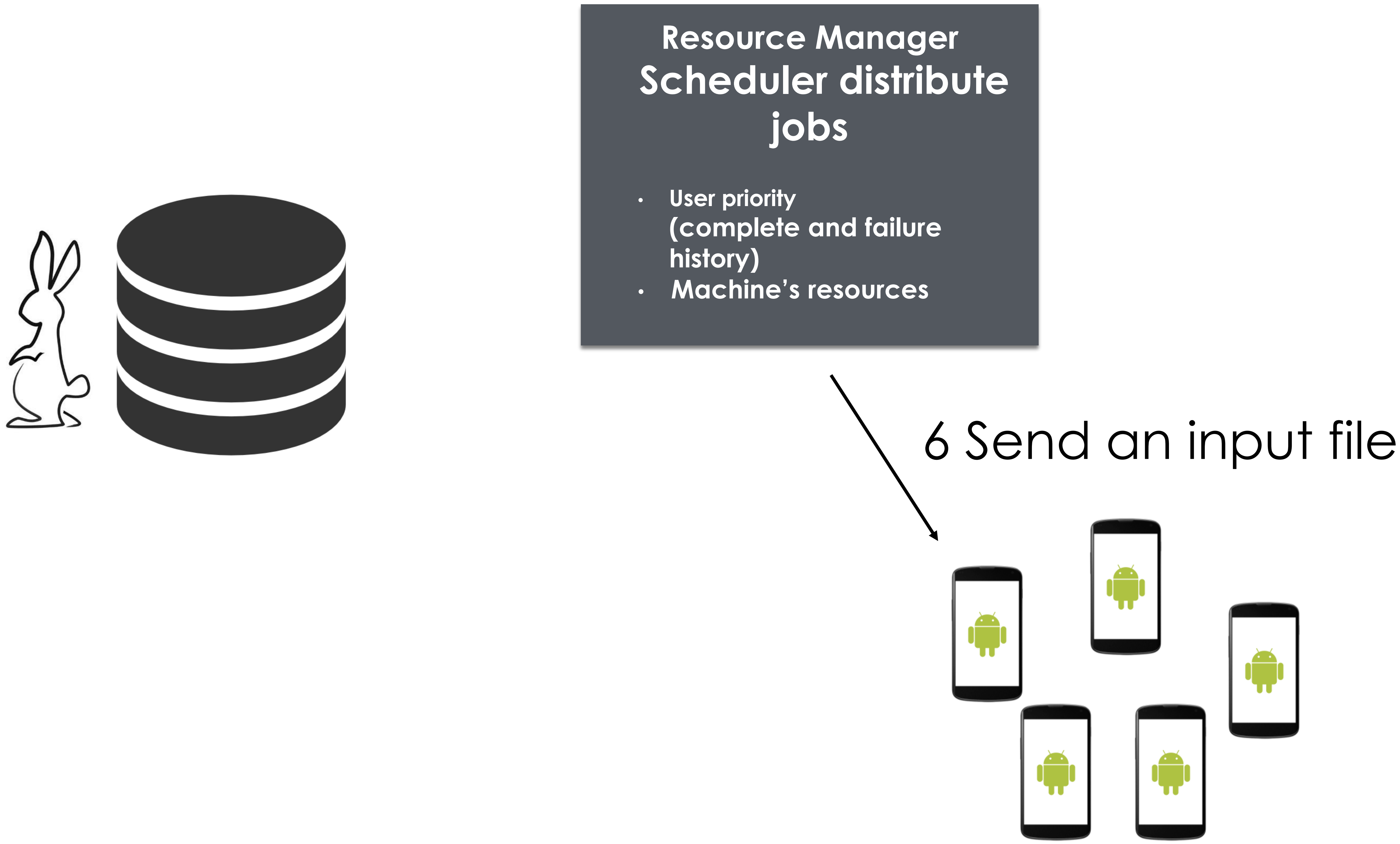
Resource Manager

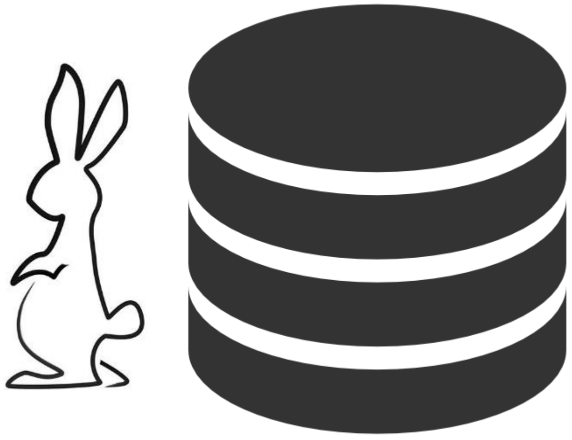
Wait for client connections

3 Await input data and Partition tasks upon receiving


Resource Manager
Wait for client connections
Register
4 Collect user information
Install the application


Resource Manager
Wait for clients
5 Sent request and resource status
User click 'run'


Resource Manager
Scheduler distribute jobs
User priority (complete and failure history)
Machine's resources
6 Send an input file

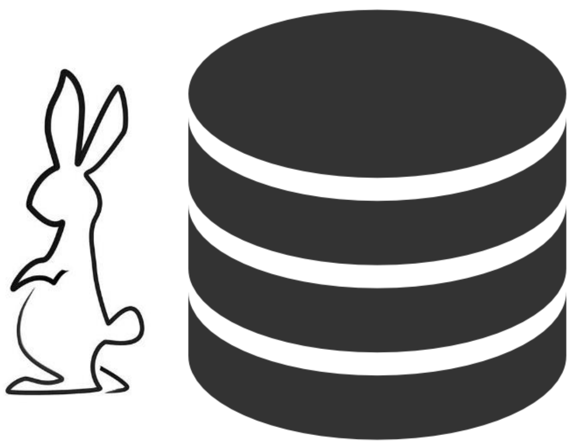


Resource Manager

Compute …

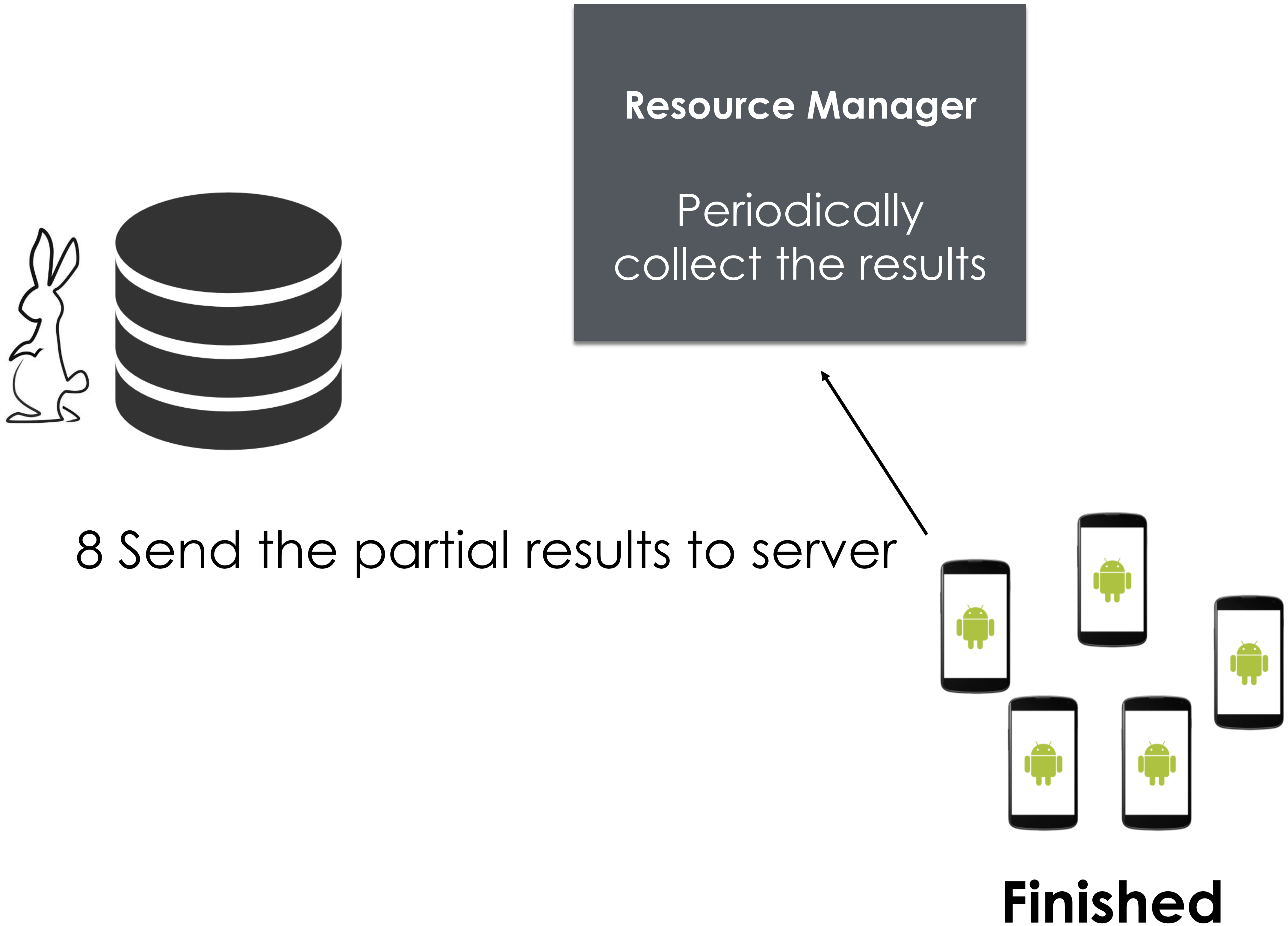
Resource Manager

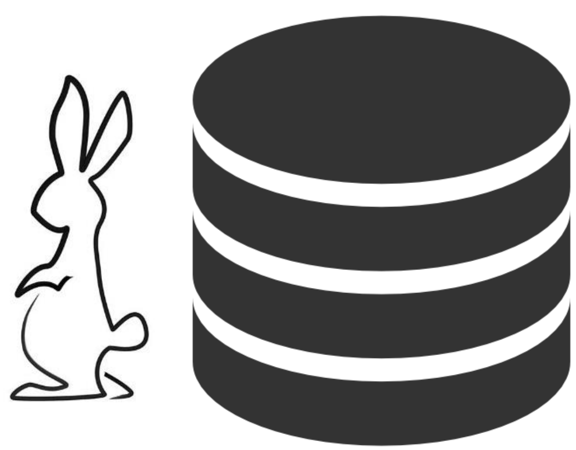
Periodically
Probe the devices



7 Send heartbeat requests to track jobs

If fail to receive heartbeats,
reschedule the old tasks


Resource Manager
Periodically collect the results
8 Send the partial results to server
Finished



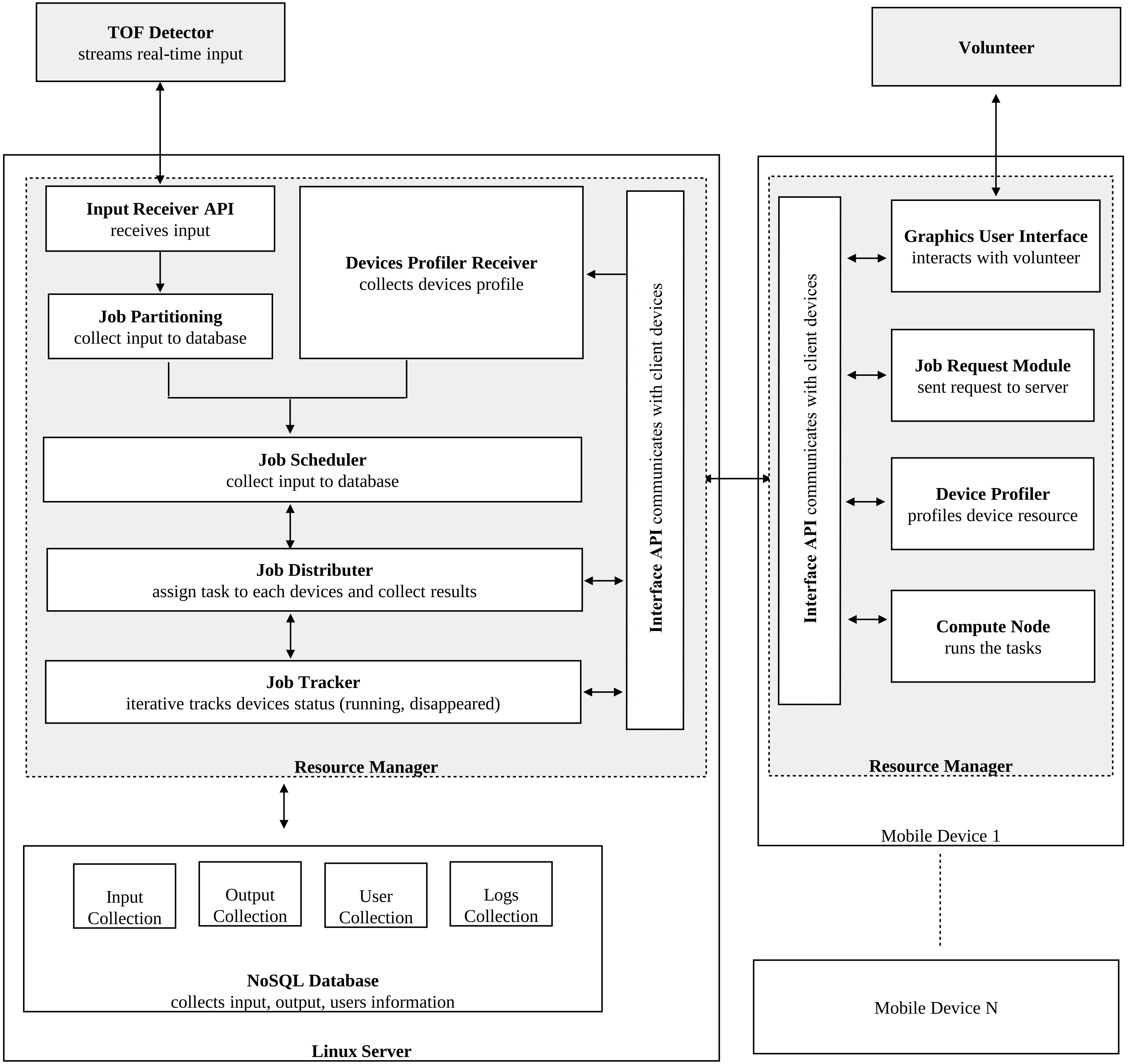
Resource Manager

Compile and Submit results

9 collect user's computation time/failure

Finished


Resource Manager
Compile and Submit results
10 Store results persistently
ALICE DB



TOF Detector
streams real-time input
Volunteer
Input Receiver API
receives input
Devices Profiler Receiver
collects devices profile
Job Partitioning
collect input to database
Job Scheduler
collect input to database
Job Distributer
assign task to each devices and collect results
Job Tracker
iterative tracks devices status (running, disappeared)
Interface API communicates with client devices
Resource Manager
Input Collection
Output Collection
User Collection
Logs Collection
NoSQL Database
collects input, output, users information
Linux Server
Interface API communicates with client devices
Graphics User Interface
interacts with volunteer
Job Request Module
sent request to server
Device Profiler
profiles device resource
Compute Node
runs the tasks
Resource Manager
Mobile Device 1
Mobile Device N

Challenges

Power - There is only a limited amount of power available on a mobile device at any given time.

Platform - Distributed applications targeted at mobile platforms have to run on heterogeneous systems using a variety of hardware, OS and libraries.

Network - A mobile device is often intermittently connected to a network and connects to a variety of networks including cellular, data, Wi-Fi.

User concerns - Mobile device owners may be concerned with battery drainage and connectivity charges.

Plans

Power – Only activate computations when devices is being charged (Offline ALICE applications only)

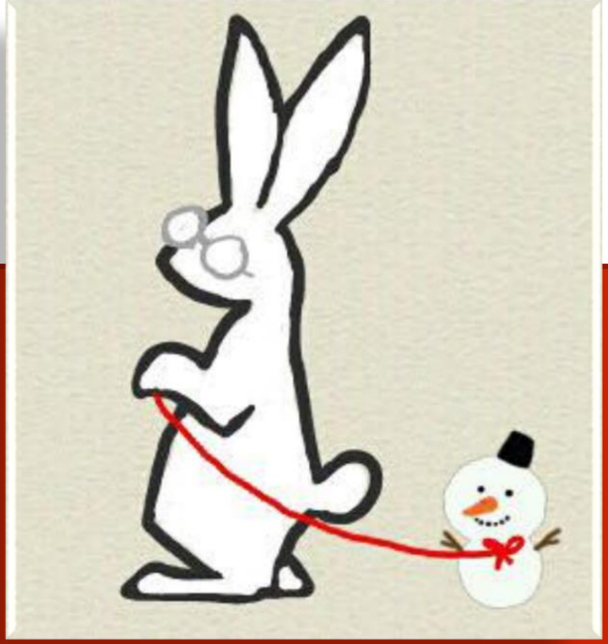
Platform – Android platform only (for now)

Network – Transfer data over WiFi only. If offline, cache the results on the phone

User concerns – Rewarding schemes will be implemented as a service to motivate users

Gamification on Social networks

Let’s share the wonderland of Sciences to the general public



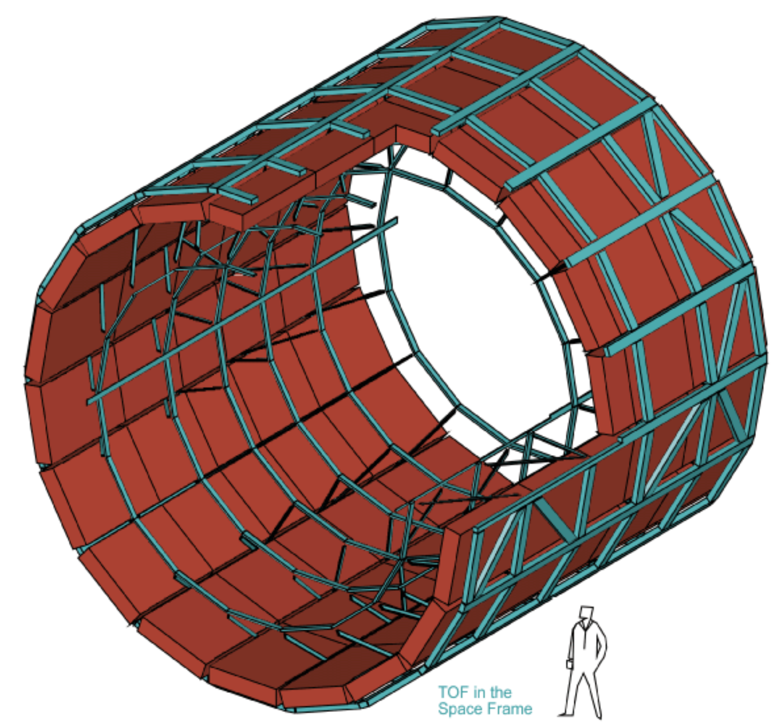
Happy Holidays

References

- Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D., 2002 , “SETI@home: an experiment in public-resource computing”, **Co mmunications of the ACM**, , pp. 56-61.
- E. Marinelli., 2009, “Hyrax: cloud computing on mobile devices using MapReduce.” **Carnegie-mellon university Pittsburgh PA school of co mputer science**.
- J. Baldassari, D.Finkel and D. Toth, 2006, “SLINC: A Framework for volu nteer computing”, **the 18th IASTED International Conference on Parall el and Distributed Computing and Systems**
- David P. Anderson, 2004, “BOINC: A System for Public-Resource Com puting and Storage”, **5th IEEE/ACM International Workshop on Grid C omputing**
- http://boinc.bakerlab.org/

TOF Detector

- A particle detector which can discriminate between a lighter and a heavier elementary particle of same momentum using their time of flight between two scintillators.
- Consists of inspecting a large number of events (>10^7)
- Detecting a maximum and then merging all the results together

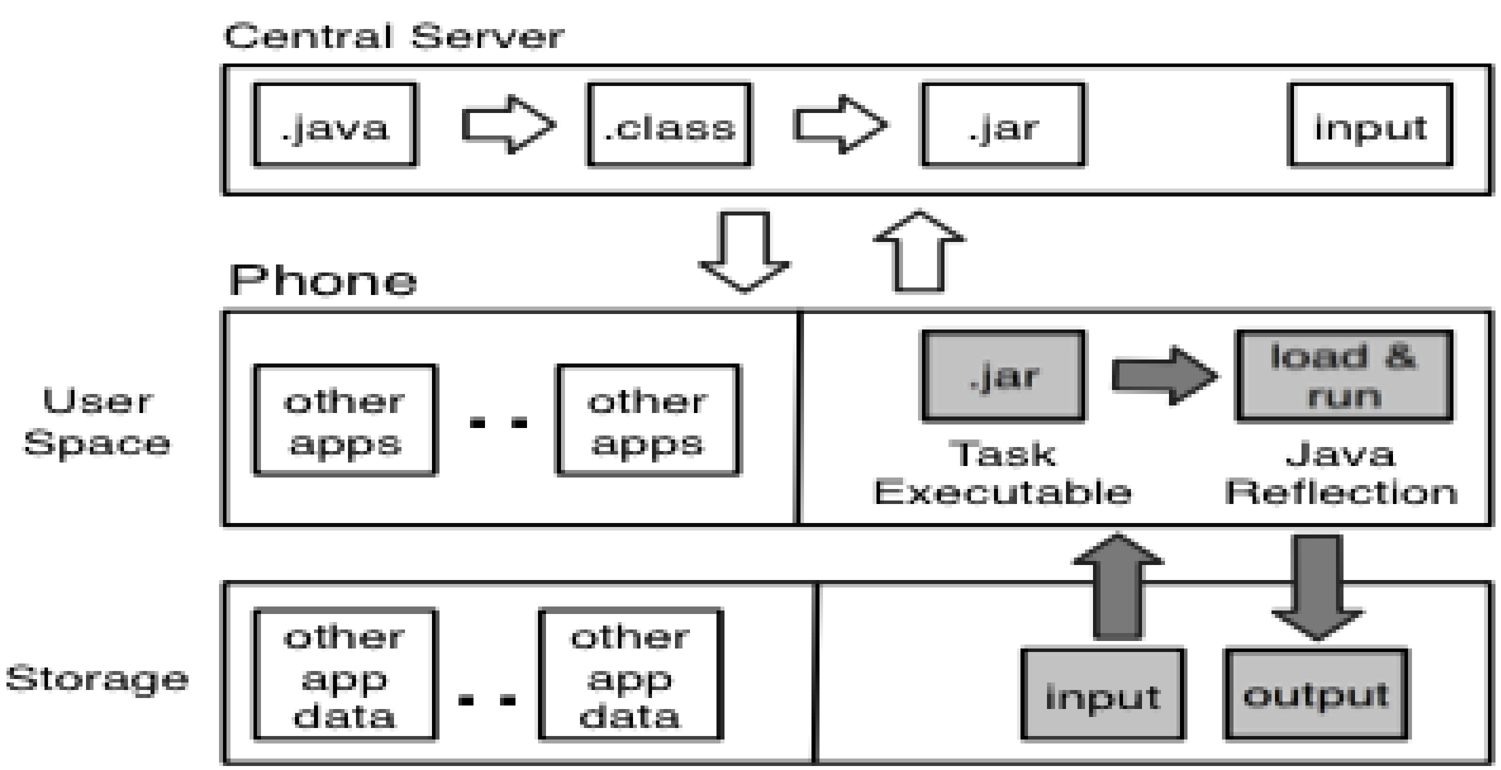


CWC: Computing While Charging

- **Goal:** To find an alternative way to run tasks by using the idle cycle of smartphone
- Focus on energy-efficient and cost-effective
  - (a) profile a charging behaviours of real phone owners
  - (b) Implement android application that provide simple task migration, interrupted and resume task executions
  - (c) deploy a prototype of CWC and evaluate the result (with 18 Android smartphones)

CWC: Computing While Charging

- a central server partitions a large input file into smaller pieces
- transmits the input partitions (together with the executable that processes the input) to the smartphones in CWC.
- Upon receiving the executable and the corresponding input, the phones execute the task in parallel
- Phones return results to the central server when they finish executing the task.
- The central server performs a logical aggregation of the returned results



CWC: Computing While Charging

Algorithm 1 Greedy Packing Algorithm

```
L : sorted list in decreasing order of execution time
C : bin capacity
repeat
  find the first item in L that can fit in any opened bin
  if such an item exists then
    pack the item in the bin with min. height
    if the item was packed as a whole then
      remove it from L
    else
      insert its remaining input in L
      re-sort L
    end if
  else
    if there are un-opened bins then
      open the best bin for the largest item in L
      pack the item in the opened bin
      if the item was packed as a whole then
        remove it from L
      else
        insert its remaining input in L
        re-sort L
      end if
    else
      cannot open any more bins
      cannot finish packing with C
    end if
  end if
until all jobs are packed
```

CBP: Complementary bin packing

$$E_j * b_i + x * (b_i + c_{ij})$$

E_j is the size (in KB) of job j's executable

b_i is the time that it takes phone i to receive 1 KB of data from the server

c_{ij} is the time that it takes for phone i to execute the job j on 1 KB of input data.