

AndroLyzeLab

[A scalable android application package analyzer]



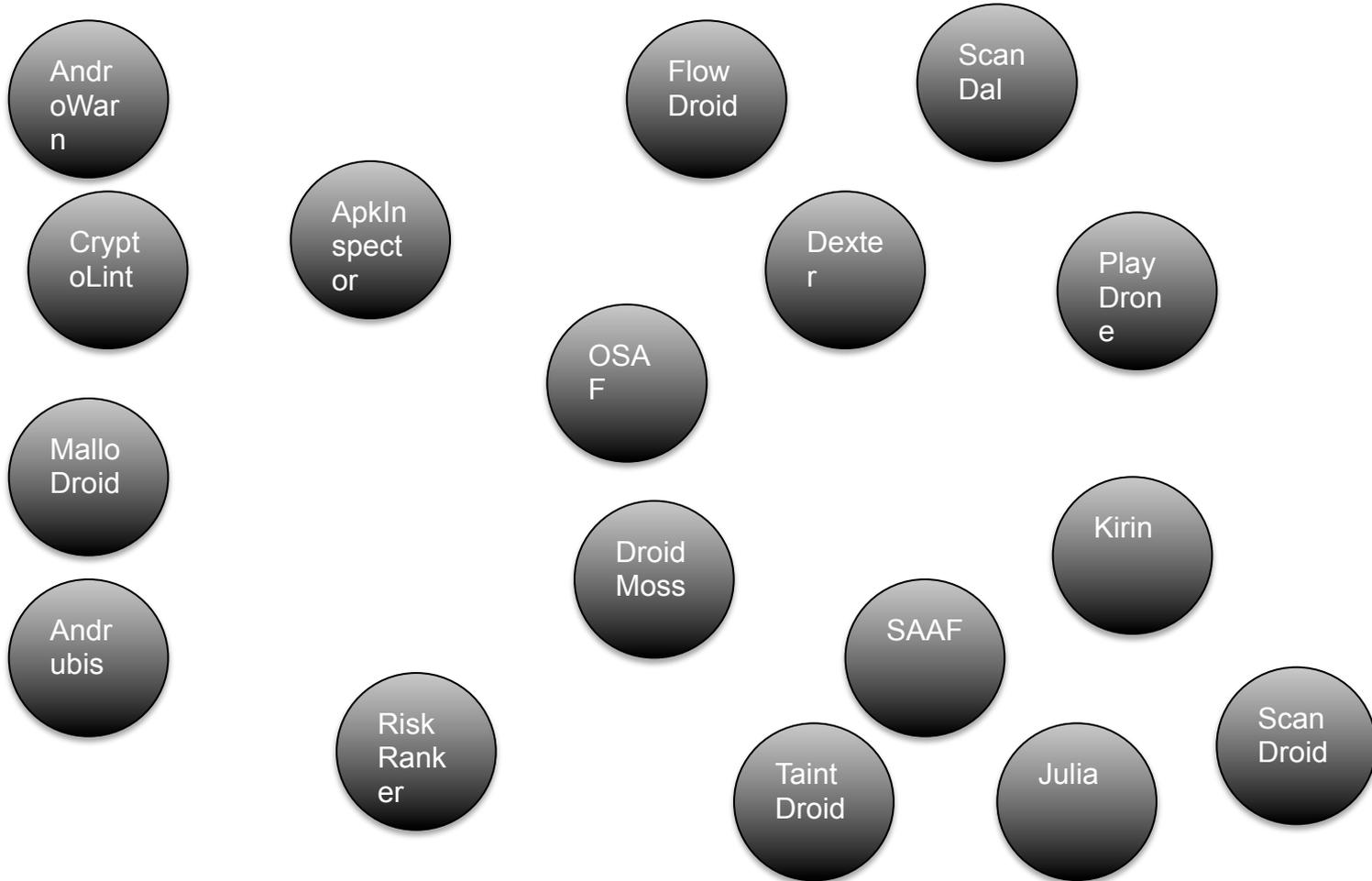
Overview

1. Intro
2. Related work
3. General overview of AndroLyzeLab
4. Typical workflow
5. Scripts
6. Storage
7. Experiments
8. Future work

Intro

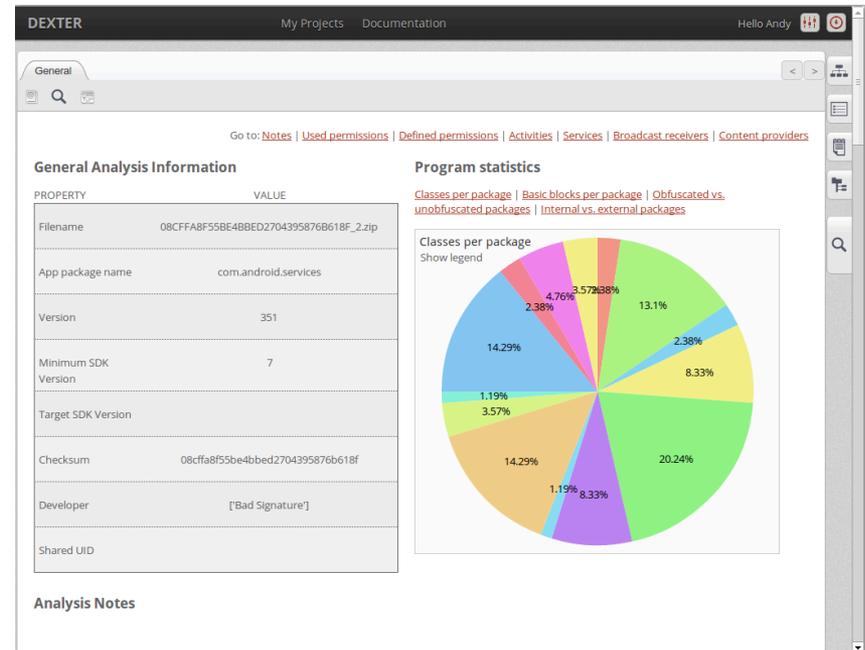
- Usage of mobile devices → sensitive data
 - Address book, mail, chats, pictures, documents, metadata
- Can we trust applications?
 - Information leakage
 - Malware
- Android:
 - market share: 78,4 % (2nd quarter 2013)
 - 900 million devices activated since 2008
 - +1,5 million per day
 - Sophos: exponential growth in malware

Related work

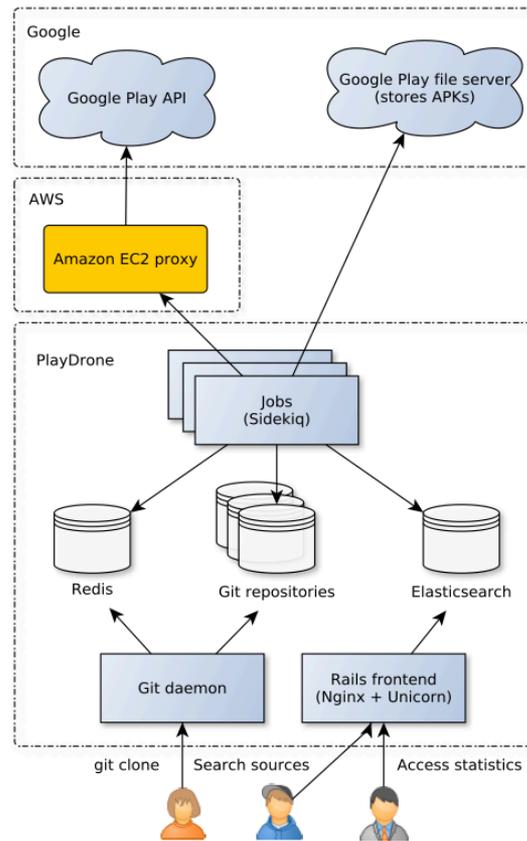


Related work: Dexter

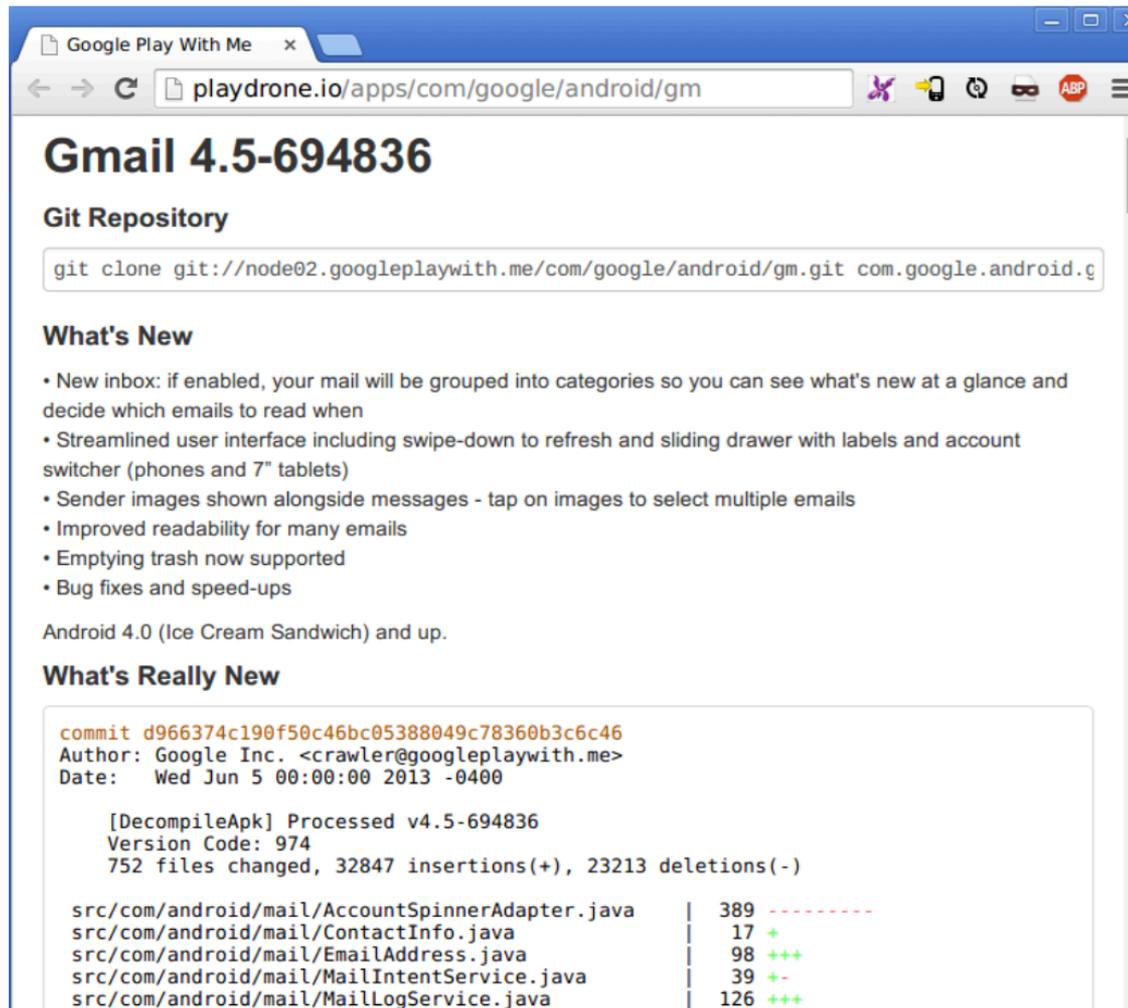
- Static android analysis tool
- Closed source, web platform
- Disassembly, decompilation
- General information and statistics
- Permissions
- UML-class diagrams
- Search
- Tagging-system
- Share results with other users
- Scalable?



Related work: PlayDrone



Related work: PlayDrone



The screenshot shows a web browser window with the address bar displaying `playdrone.io/apps/com/google/android/gm`. The page title is "Gmail 4.5-694836". Below the title, there is a "Git Repository" section with a code block containing the command: `git clone git://node02.googleplaywith.me/com/google/android/gm.git com.google.android.g`. The "What's New" section lists several updates: a new inbox grouping feature, a streamlined user interface, sender images, improved readability, trash emptying, and bug fixes. Below this, it mentions "Android 4.0 (Ice Cream Sandwich) and up." The "What's Really New" section shows a commit hash `d966374c190f50c46bc05388049c78360b3c6c46` by Google Inc. with a date of "Wed Jun 5 00:00:00 2013 -0400". It also includes a decompile log for version 4.5-694836, showing 752 files changed, 32847 insertions, and 23213 deletions. A table at the bottom lists file changes with their respective counts and status indicators.

Gmail 4.5-694836

Git Repository

```
git clone git://node02.googleplaywith.me/com/google/android/gm.git com.google.android.g
```

What's New

- New inbox: if enabled, your mail will be grouped into categories so you can see what's new at a glance and decide which emails to read when
- Streamlined user interface including swipe-down to refresh and sliding drawer with labels and account switcher (phones and 7" tablets)
- Sender images shown alongside messages - tap on images to select multiple emails
- Improved readability for many emails
- Emptying trash now supported
- Bug fixes and speed-ups

Android 4.0 (Ice Cream Sandwich) and up.

What's Really New

```
commit d966374c190f50c46bc05388049c78360b3c6c46
Author: Google Inc. <crawler@googleplaywith.me>
Date:   Wed Jun 5 00:00:00 2013 -0400

[DecompileApk] Processed v4.5-694836
Version Code: 974
752 files changed, 32847 insertions(+), 23213 deletions(-)

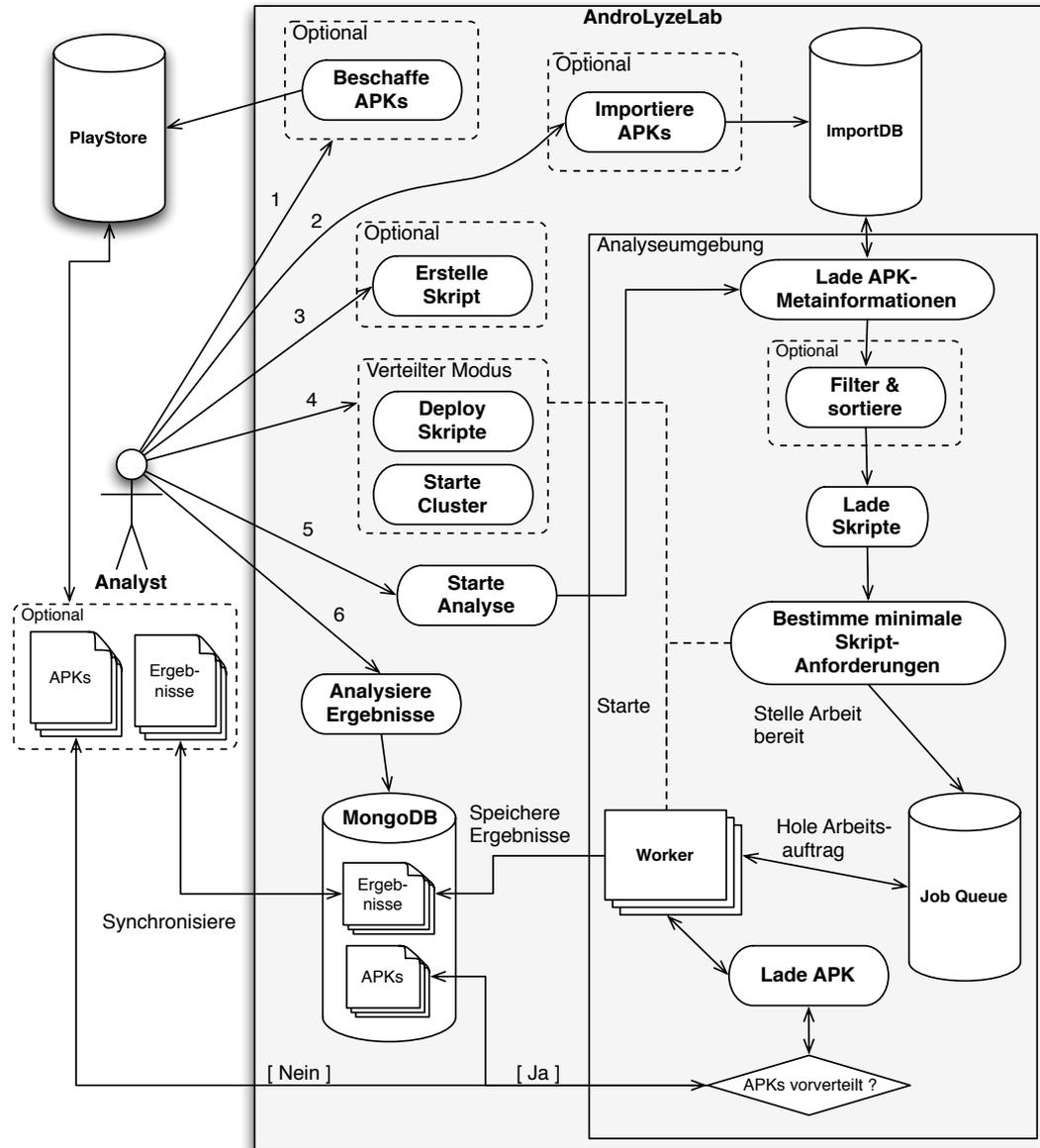
src/com/android/mail/AccountSpinnerAdapter.java | 389 - - - - -
src/com/android/mail/ContactInfo.java           | 17 +
src/com/android/mail/EmailAddress.java          | 98 +++
src/com/android/mail/MailIntentService.java    | 39 +-
src/com/android/mail/MailLogService.java       | 126 +++
```

Overview

- Main purpose?
 - Analyze APKs
- How?
 - Use analysis functionality of androguard
 - Includes disassembler and *native* decompiler (DAD)
- From where do we get them?
 - Use *PlaystoreCrawler* to download APKs from Google Play Store
 - Update whole APK collection
- How do we organize many APKs?
 - Import metadata into database (sqlite3)
 - Adds filtering capabilities -> Only analyze subset matching filter
- How do we store analysis results?
 - Use MongoDB
 - Schema free database (NoSQL) for dynamic results
 - Filesystem storage optional

Overview #2

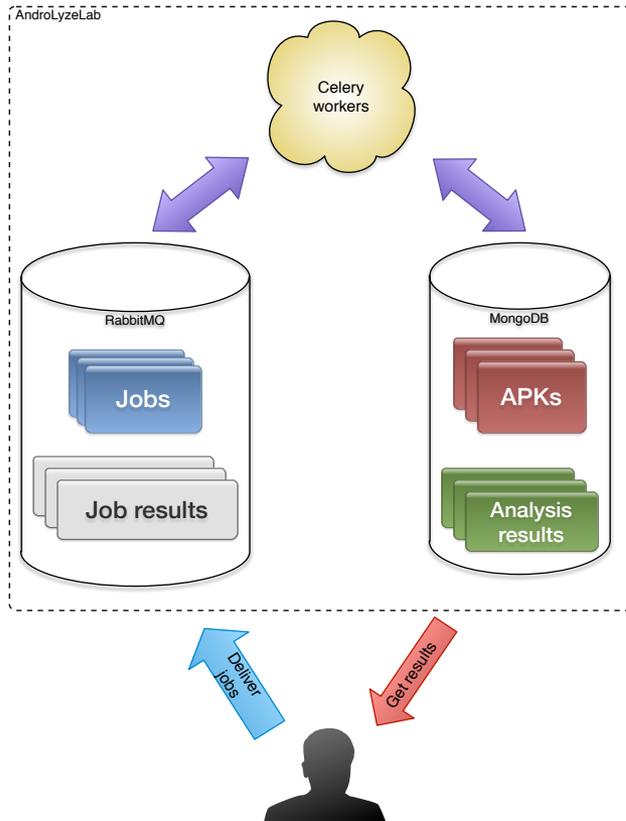
- How to extend it?
 - Use AndroLyzeLab script framework (python2)
 - Think about a result structure
 - Analyze with androguard
 - Use logging system to store results
- How to evaluate results?
 - Query MongoDB
 - Use integrated query builder
 - Complex: MapReduce & JavaScript



Parallelization

- Determine minimum script requirements
- Open APK with androguard
- Run script(s)
- Store result(s) in mongoDB
- Use processes due to CPython Global Interpreter Lock
- Granularity
 - 1 Process = APK with all scripts
 - Due to androguard APK open overhead
- Local
 - Use producer-worker pattern with queues

Parallelization: Distributed



- Message-oriented middleware
- Publish jobs in queue
 - Serialize APK
 - Or APK ID (for mongoDB)
- Store results in mongoDB
- Store result IDs in result queue
- Fetch results from mongoDB

Design: Message format

Argument	Type
Scripts	List<String> // package name
Script hashes	List<String> // sha256
APK / APK ID	String
Is_hash	Boolean
APK metadata	FastApk

Design: Properties

- Fault tolerance:
 - Broker network error -> retry until tasks published
 - MongoDB network error -> retry until results stored
 - Node crash -> Job still in queue
- Dynamic scheduling
- Add/remove nodes by need

Scripts: Logging

```
com.zoner.android.antivirus_1.9.0_ShowLoggingFuncs.json UNREGISTERED
1 {
2   "apk meta": {
3     "package name": "com.zoner.android.antivirus",
4     "version name": "1.9.0",
5     "sha256": "88e53942dffe88aa1bbaf374b55d761",
6     "import date": "2014-12-03T10:15:20.701880",
7     "path": "/Users/nils/master/it_sec2/worksp",
8     "tag": "zoner"
9   },
10  "script meta": {
11    "name": "ShowLoggingFuncs",
12    "sha256": "3f8a83e9abcaf32de59260df690be8b",
13    "analysis date": "2014-12-07T19:25:31.5884",
14    "version": "0.1"
15  },
16  "demo": {
17    "logged": {
18      "normal": "some value",
19      "bool": true,
20      "enum": [
21        "list element"
22      ]
23    },
24    "unlogged": {
25      "normal": null,
26      "bool": false,
27      "enum": []
28    }
29  }
30 }
```

Static structure

```
ShowLoggingFuncs.py UNREGISTERED
1 from androlyzlab.model.script.AndroScript import AndroScript
2
3 CAT_UNLOGGED = "demo", "unlogged"
4 CAT_LOGGED = "demo", "logged"
5
6 class ShowLoggingFuncs(AndroScript):
7
8     VERSION = "0.1"
9
10    def _analyze(self, apk, dalvik_vm_format, vm_analysis,
11                gvm_analysis, *args, **kwargs):
12        res = self.res
13
14        # register structure
15        res.register_keys(["normal"], *CAT_LOGGED)
16        res.register_keys(["normal"], *CAT_UNLOGGED)
17        res.register_bool_keys(["bool"], *CAT_LOGGED)
18        res.register_bool_keys(["bool"], *CAT_UNLOGGED)
19        res.register_enum_keys(["enum"], *CAT_LOGGED)
20        res.register_enum_keys(["enum"], *CAT_UNLOGGED)
21
22        # log
23        res.log("normal", "some value", *CAT_LOGGED)
24        res.log_true("bool", *CAT_LOGGED)
25        res.log_append_to_enum("enum", "list element", *CAT_LOGGED)
26
27    if __name__ == '__main__':
28        for res in AndroScript.test(ShowLoggingFuncs, ["a2dp.Vol.apk"]):
29            print res
30            print res.write_to_json()
```

Dynamic structure

Scripts: ClassDetails

```
ClassDetails.py UNREGISTERED
9  from androlyzelab.model.script.AndroScript import AndroScript
10
11 # categories
12 CAT_CLASS_DETAILS = "class details"
13 CAT_METHODS = "methods"
14 CAT_FIELDS = "fields"
15
16 class ClassDetails(AndroScript):
17     ''' Retrieve all classes and their methods and fields '''
18
19     VERSION = "0.1"
20
21     def _analyze(self, apk, dalvik_vm_format, vm_analysis, gvm_analysis, *args, **kwargs):
22         res = self.res
23
24         # dvm stuff
25         # list<ClassDefItem>
26         classes = dalvik_vm_format.get_classes()
27
28         # run over classes
29         for c in classes:
30             ROOT_CAT = (CAT_CLASS_DETAILS, c.name)
31             res.register_keys([CAT_METHODS, CAT_FIELDS], *ROOT_CAT)
32
33             # list<EncodedMethod>
34             methods = c.get_methods()
35             res.log(CAT_METHODS, [mn.name for mn in methods], *ROOT_CAT)
36
37             # list<EncodedField>
38             fields = c.get_fields()
39             res.log(CAT_FIELDS, [fn.name for fn in fields], *ROOT_CAT)
40
41             #####
42             #---Options
43             #####
44
45         def needs_dalvik_vm_format(self):
46             return True
47
48     if __name__ == '__main__':
49         for res in AndroScript.test(ClassDetails, ["../testenv/apks/a2dp.Vol.apk"]):
50             print res
51             print res.write_to_json()

```

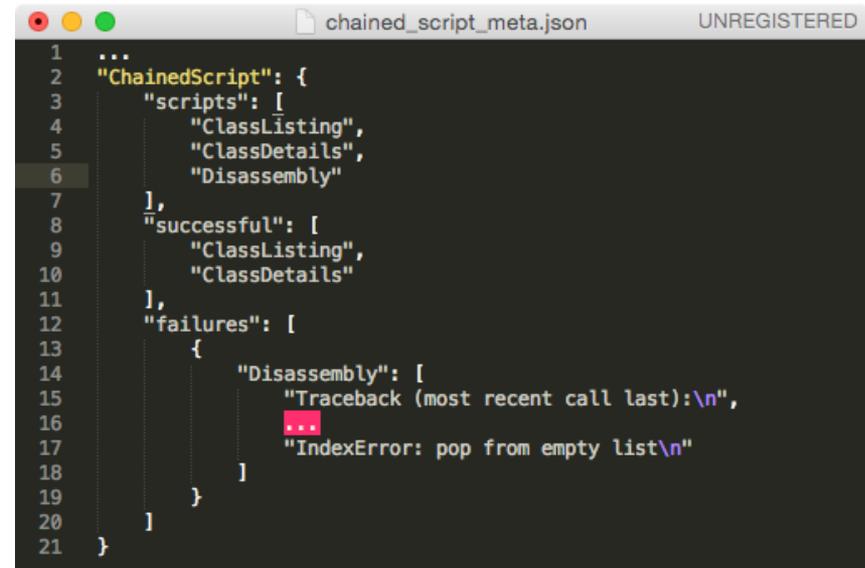
Line 51, Column 34 Spaces: 4 Python

Scripts: Options

```
ScriptTemplate.py UNREGISTERED
30
31 #####
32 #—Script requirements
33 #####
34
35 def needs_dalvik_vm_format(self):           Disassemble
36     return False
37
38 def needs_vmanalysis(self):                 Analyze bytecode
39     return False
40
41 def needs_gvmanalysis(self):                CFG → cross refs.
42     return False
43
44 def needs_xref(self):                       Function refs.
45     return False
46
47 def needs_dref(self):                       Data refs.
48     return False
49
50 #####
51 #—Options
52 #####
53
54 def create_script_stats(self):              Show timings
55     return False
56
57 def is_big_res(self):                       Result ≥ 16MB → gridFS
58     return False
59
```

Scripts: Chaining

- Do it like in *software engineering*
 - Keep it modular! And chain by need!
- Options:
 - Log errors of chained scripts
 - Skip/continue if one script failed

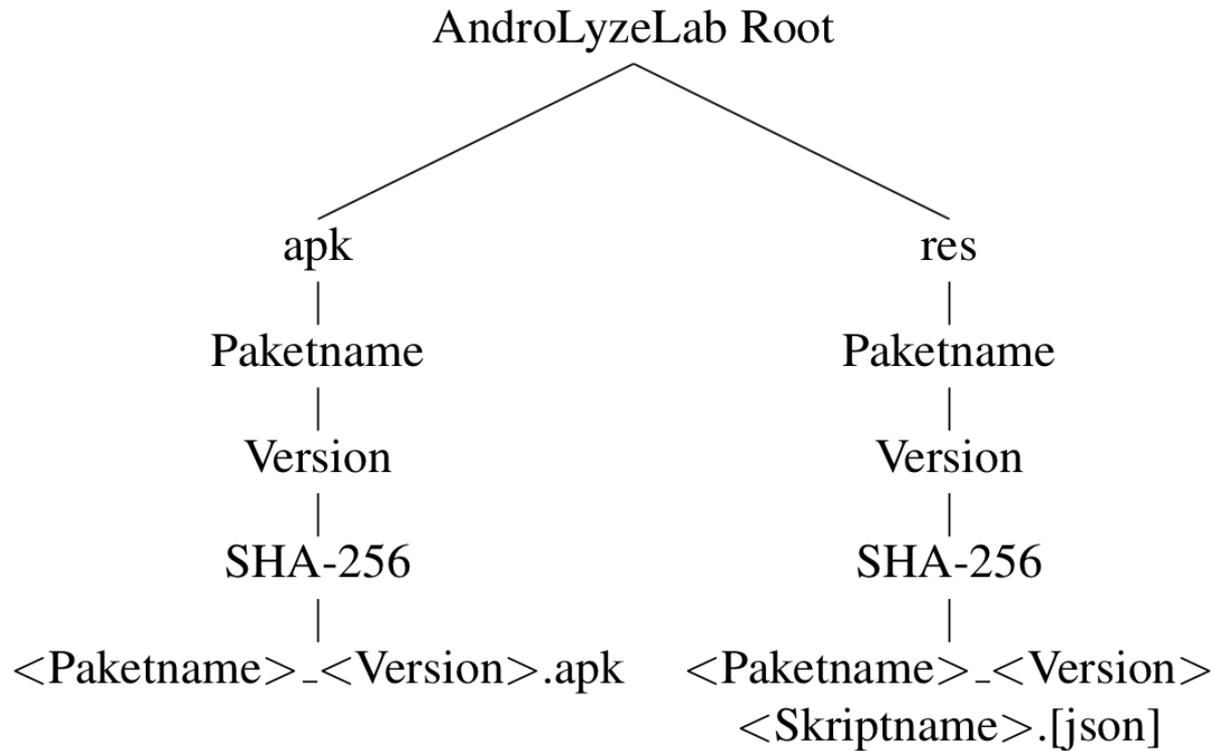


```
1  ...
2  "ChainedScript": {
3    "scripts": [
4      "ClassListing",
5      "ClassDetails",
6      "Disassembly"
7    ],
8    "successful": [
9      "ClassListing",
10     "ClassDetails"
11   ],
12   "failures": [
13     {
14       "Disassembly": [
15         "Traceback (most recent call last):\n",
16         "...
17         "IndexError: pop from empty list\n"
18       ]
19     }
20   ]
21 }
```

Scripts: Built-in

1. Manifest
 1. Activities
 2. Services
 3. Files
 4. Intents
 5. Libs
 6. Permissions
 7. ContentProviders
2. Bytecode
 1. ClassDetails
 2. ClassListing
3. Source code
 1. CodePermissions
 2. Decompile(Text)
4. Misc
 1. SSL
 2. GraphAnalysis
 3. GVMAAnalysisExample
 4. AnalyzeFrameworks

Storage: Filesystem



Storage: MongoDB

1. Storage via BSON
2. MongoDB
 1. Authentication: plain vs. credentials vs. credentials + SSL/TLS
 2. Document limitation: 16MB
 1. Use gridfs!
 2. Split files into chunks
 3. Stores files binary with meta data
 4. Can still query metadata
 5. Store big files:
 1. Method call graph
 2. Decompiled code

Querying MongoDB : Samples

1. `find({'$and': [{u'SSL.url_connection_https': True}, {u'hostname_verifier_allow_all': True}], 'script meta.name': u'SSL'}, {'_id': 0})`
2. `find({u'Listings.Frameworks': {'$ne': None}, 'script meta.name': u'AnalyzeFrameworks'}, {u'Listings.Frameworks': 1, u'apk meta.package name': 1, '_id': 0})`

Querying MongoDB : Assistant

Results view

AndroLyzeLab

AndroLyzeLab

Analysis Import **Result Query** Sync

Apk Filter (single value)

Package name: a2dp.Vol

Hash:

Version name:

Tag:

Script Filter (single value)

Name: AnalyzeFrameworks

Version:

Hash:

Checks (supply field names comma separated)

Null: apk meta.tag,otherkey.subkey

Not null:

True:

False:

Empty list:

List not empty:

Conjunction

Or And

Projection:

Include fields
 Exclude fields

Listings.Frameworks

Distinct key:

Options:

Latest entry only
 Count
 List ran scripts
 Sort by analysis date

GridFS:

GridFS
 Get raw data

Custom query:

```
{'Listings.Frameworks': 'phonegap'}
```

Query

```
AndroLyzeLab — bash — 121x5
```

```
AndroLyzeLab — bash — 130x5
```

```
nachtmaar:AndroLyzeLab nils$ ./androquery -vvvvv result -sn AnalyzeFrameworks -if "apk meta.package name" "Listings.Frameworks" --  
where-dict '{"Listings.Frameworks': 'phonegap'}'
```

User interface – CLI & GUI

The screenshot displays the AndroLyzeLab application interface. The main window is titled "AndroLyzeLab" and has tabs for "Analysis", "Import", "Result Query", and "Sync".

Apks:

Package name	Version name	Import date
a2dp.Vol	2.5.2	2014-08-04 13:14:35.1877
a3g.emysh...	2.3.3	2014-08-04 13:14:35.1836
aberl.vlc.lig...	3.10.2	2014-08-04 13:14:35.2830
abinskino.p...	2.3.9	2014-08-04 13:14:35.7547
abisogullari...	3.3	2014-08-04 13:14:35.4454
ac.lite	1.05	2014-08-04 13:14:36.6334
ackdev.com...	1.5.1	2014-08-04 13:14:36.1246
ackdev.com...	1.5.1	2014-08-04 13:14:37.0132

Scripts:

Script name	Version	Description
ContentProviders	0.1	List content providers
BroadcastReceivers	0.1	List broadcast receivers
Permissions	0.1	List the permissions
Activities	0.1	List activities
ChainedApkInfos	0.1	The same as the `ApkIn
Services	0.1	Read services from mani
Intents	0.1	Get intents
Libs	0.1	List the libraries

Results view

Results:

```
},
  "permissions": [
    "android.permission.ACCESS_COARSE_LOCATION",
    "android.permission.ACCESS_FINE_LOCATION",
    "android.permission.ACCESS_LOCATION_EXTRA_COMMANDS",
    "android.permission.ACCESS_WIFI_STATE",
    "android.permission.BLUETOOTH",
    "android.permission.BLUETOOTH_ADMIN",
    "android.permission.CHANGE_WIFI_STATE",
    "android.permission.KILL_BACKGROUND_PROCESSES",
    "android.permission.MODIFY_AUDIO_SETTINGS",
    "android.permission.READ_CONTACTS",
    "android.permission.RECEIVE_BOOT_COMPLETED",
    "android.permission.RECEIVE_SMS",
    "android.permission.RESTART_PACKAGES",
    "android.permission.WRITE_EXTERNAL_STORAGE",
    "com.android.launcher.permission.READ_SETTINGS"
  ],
  "libraries": [],
  "files": [
    "res/layout/app_list.xml",
    "res/layout/app_list_item.xml",
    "res/layout/custom_intent.xml",
    "res/layout/edit_item.xml",
  ]
}
```

Step: 1

[Show more results ...](#)

Cluster management

- Management via SSH
- Define nodes in config
- Functions:
 - Install AndroLyzeLab and configure it
 - Deploy scripts
 - Start/stop/restart nodes

Experiments: Physical system

Key	Value
Kernel	3.11.0-23-generic
RAM	4x 8GiB DIMM DDR3 Synchronous 1600 MHz (0,6 ns)
SSD	Toshiba MKNSSDCR240GB
HDD	RAID1: 2x 3TB
CPU	Intel(R) Core(TM) i7-4771 CPU @ 3.50GHz
Cores	4
Cache sizes L1/L2/L3	256KiB, 1MiB, 8MiB
Network	Ethernet Connection I217-LM 1Gbit/s

Experiments: APKs

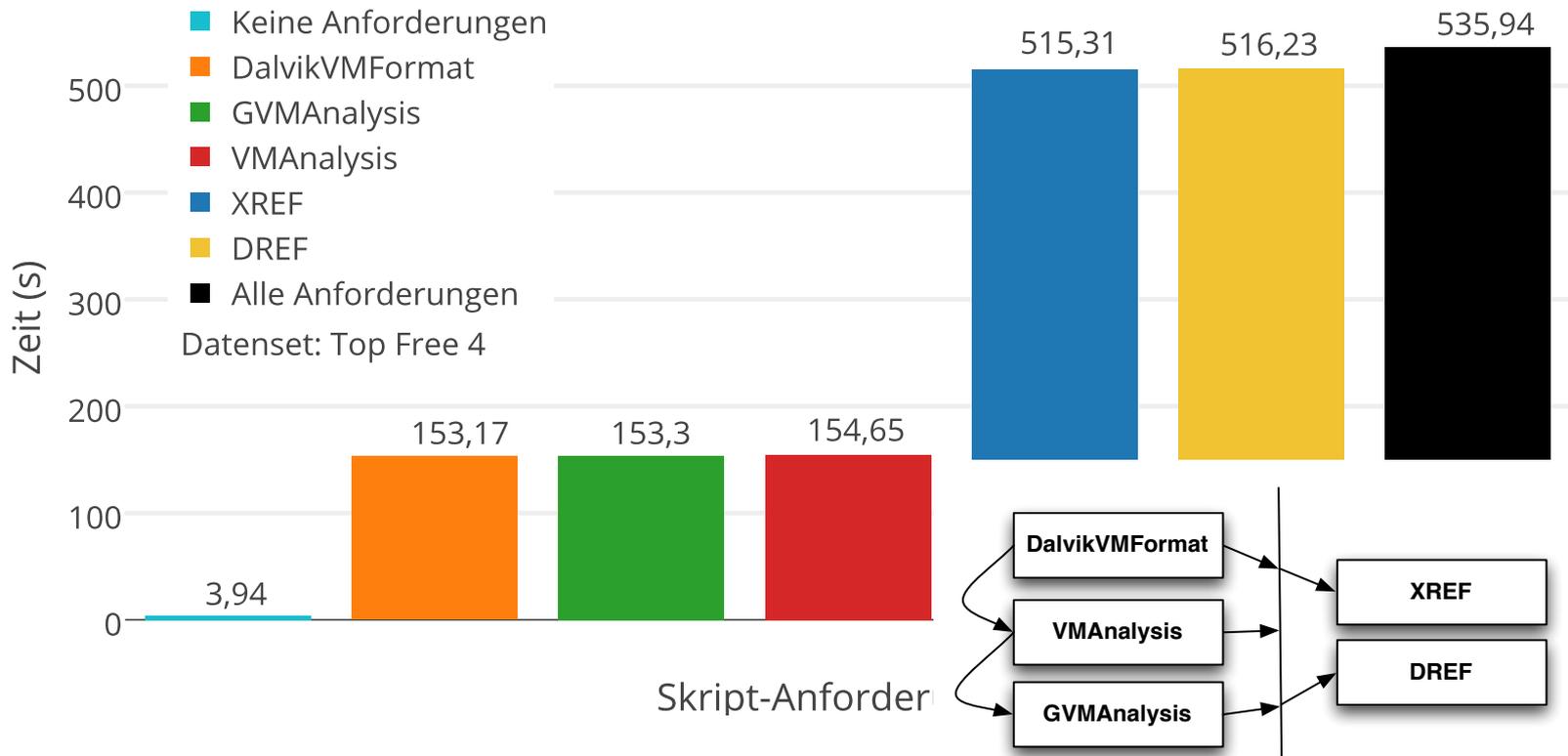
Set	Description	Count	Size (MB)
ApkSet1	Top Free 4	102	1.159
ApkSet2	Top Free 100	2.519	22.315
ApkSet3	Top Free 500	12.689	91.764

Experiments: Scripts

Set	Requirements	Scripts
Manifest	-	ChainedApkInfos, Files, Libs, Activities, Intents, ContentProviders, Services, BroadcastReceivers, Permissions
Manifest + SSL	XREF	Manifest U {SSL}
Misc1	XREF	Manifest + SSL U {ClassListing, ClassDetails, AnalyzeFrameworks, GVMAAnalysisExample}
Misc2	XREF	Misc1 U {Decompile}

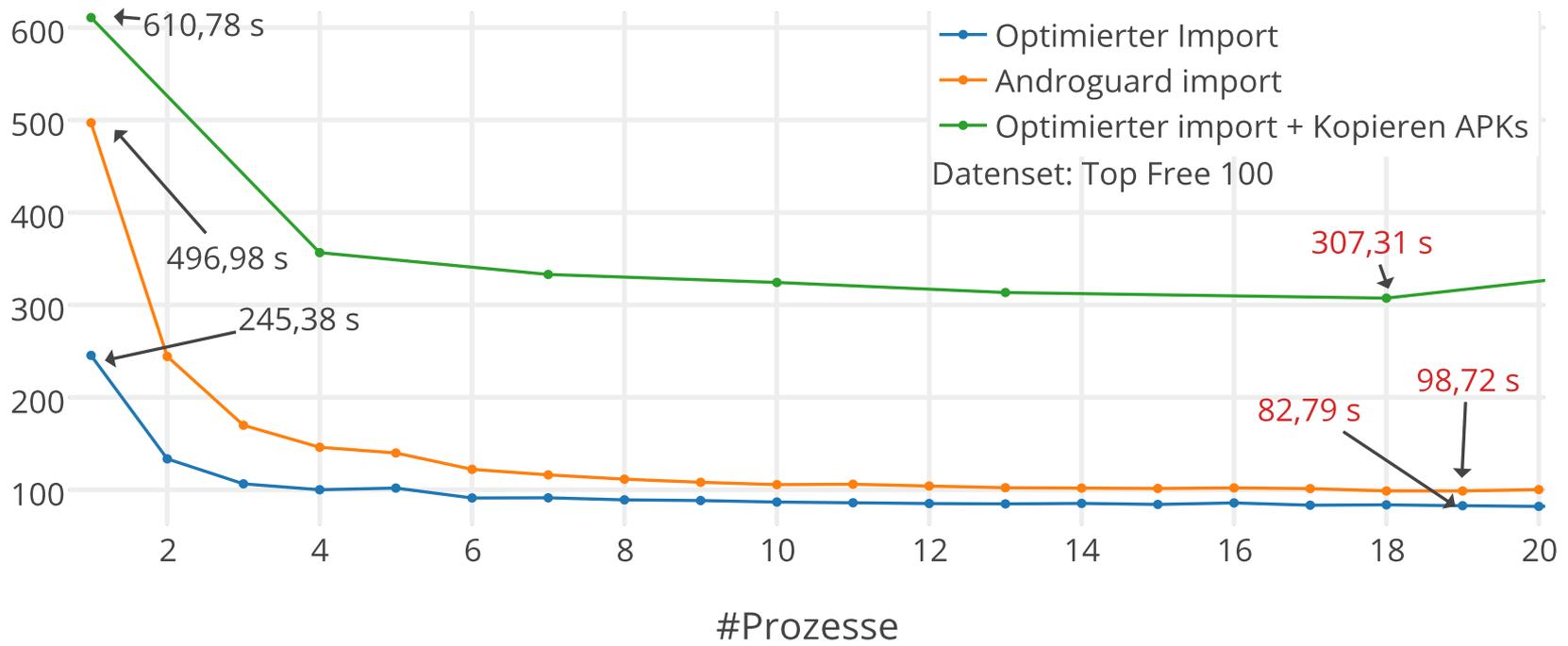
Experiments: #1

Skript-Anforderungen (androguard)



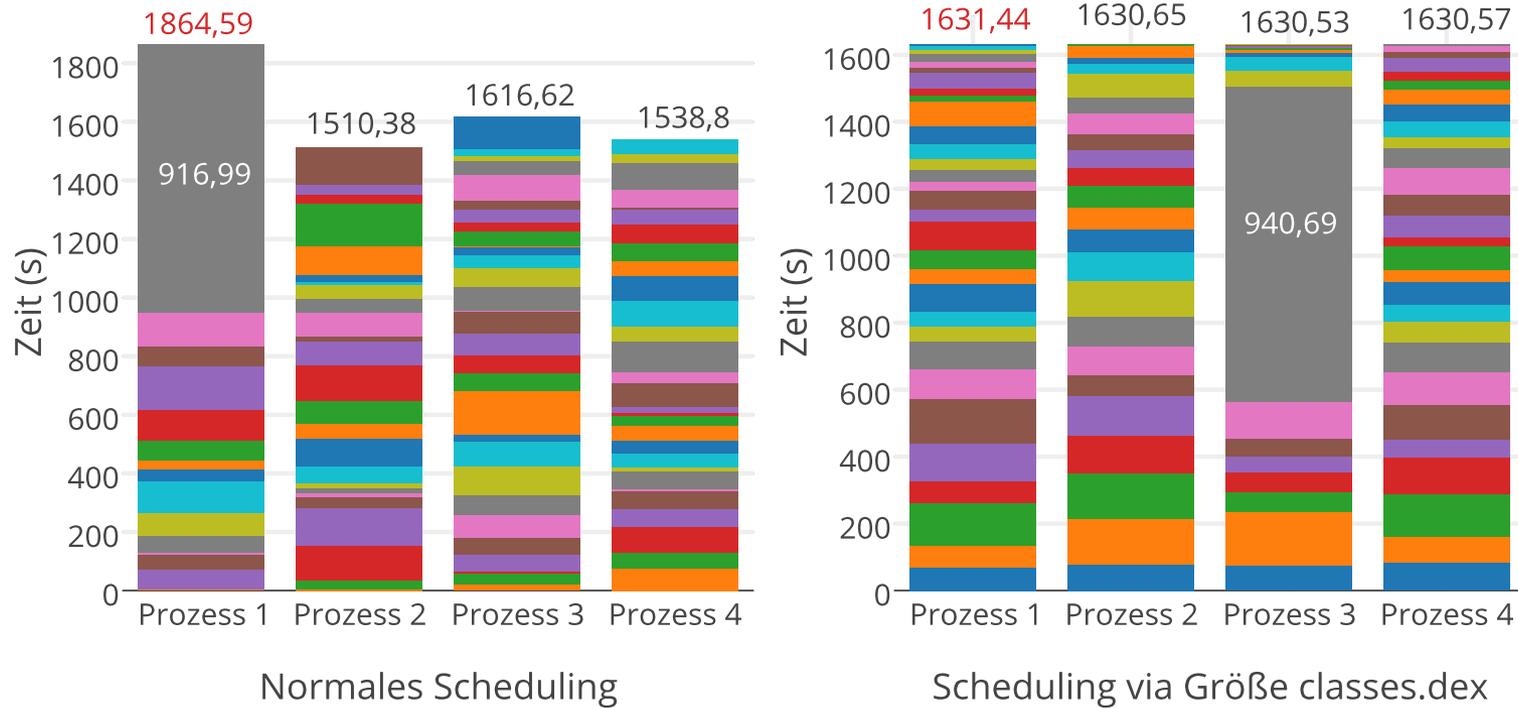
Experiments: #2

Import Experiment



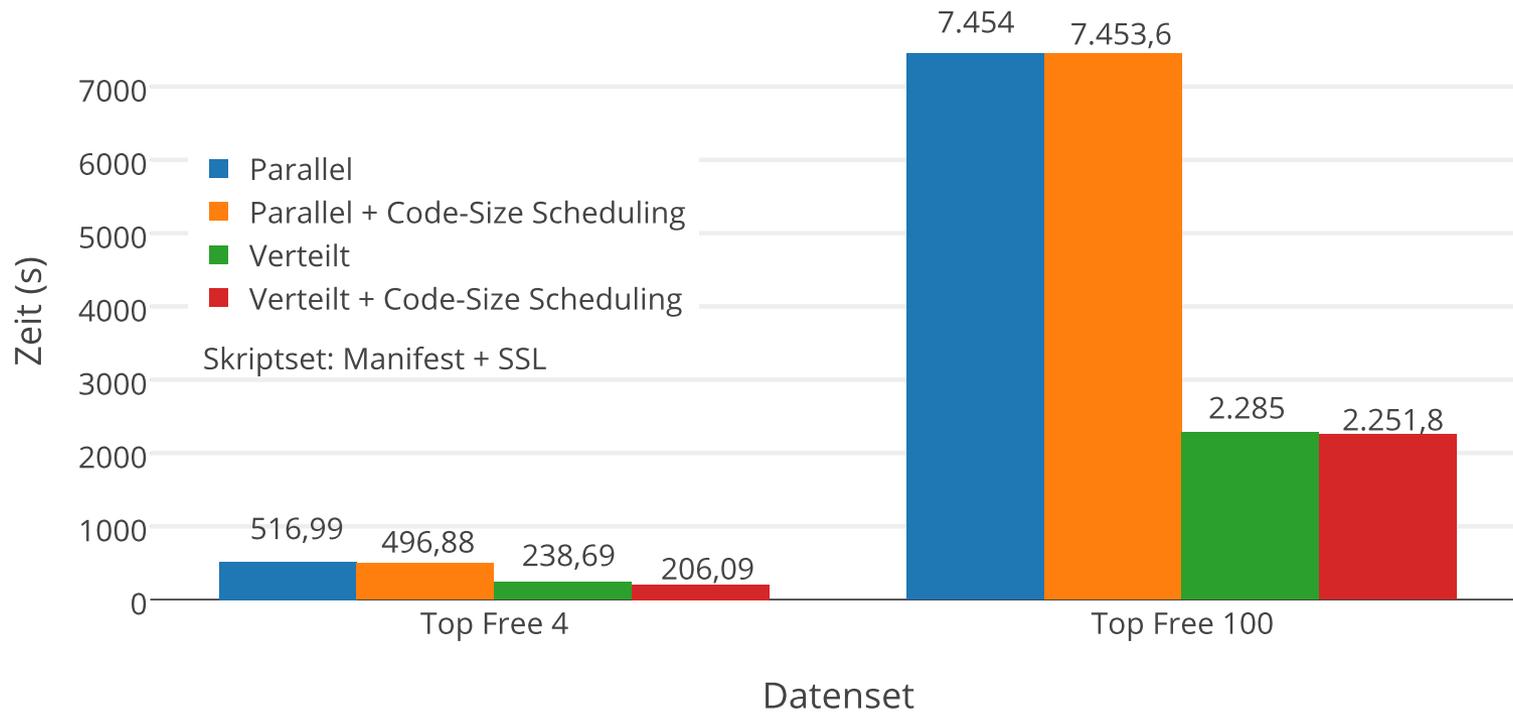
Experiments: #3

Vergleich Scheduling-Strategien



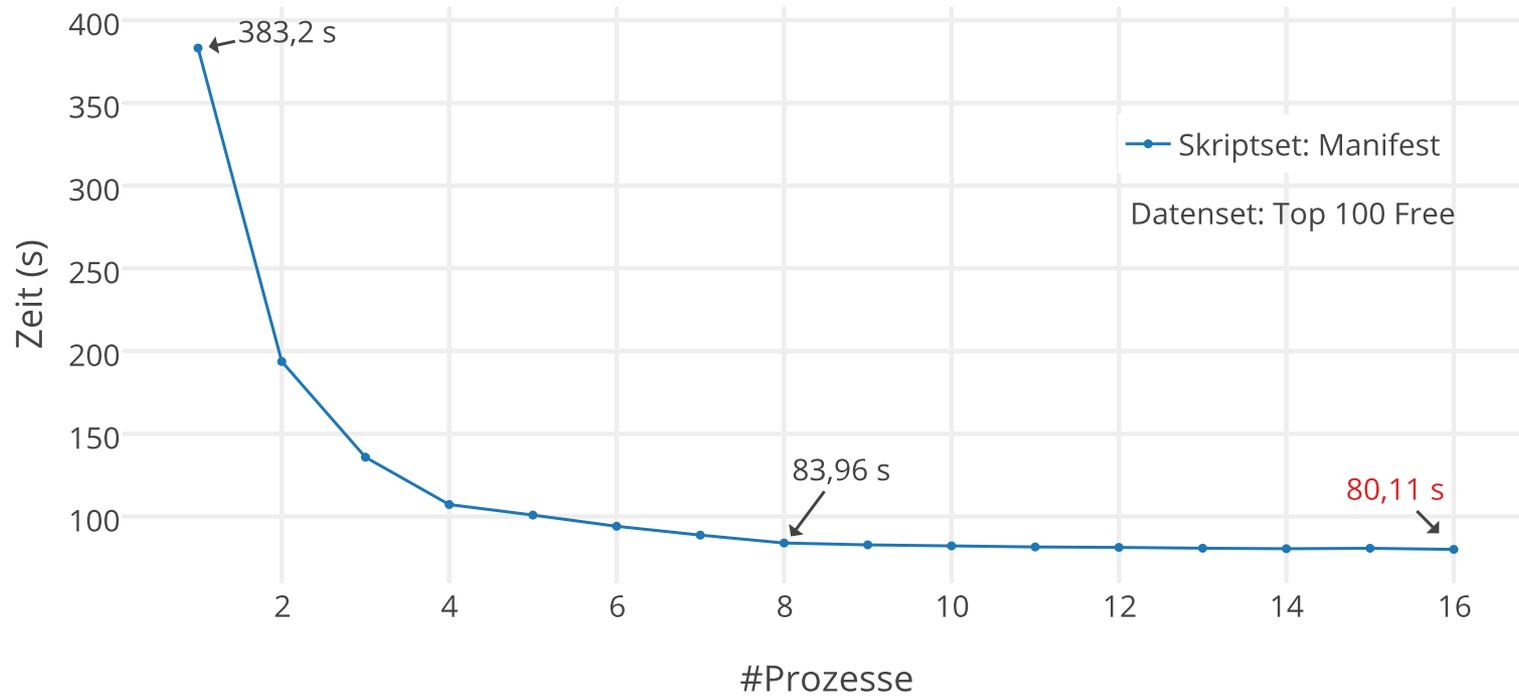
Experiments: #4

Paralleles vs. verteiltes Rechnen



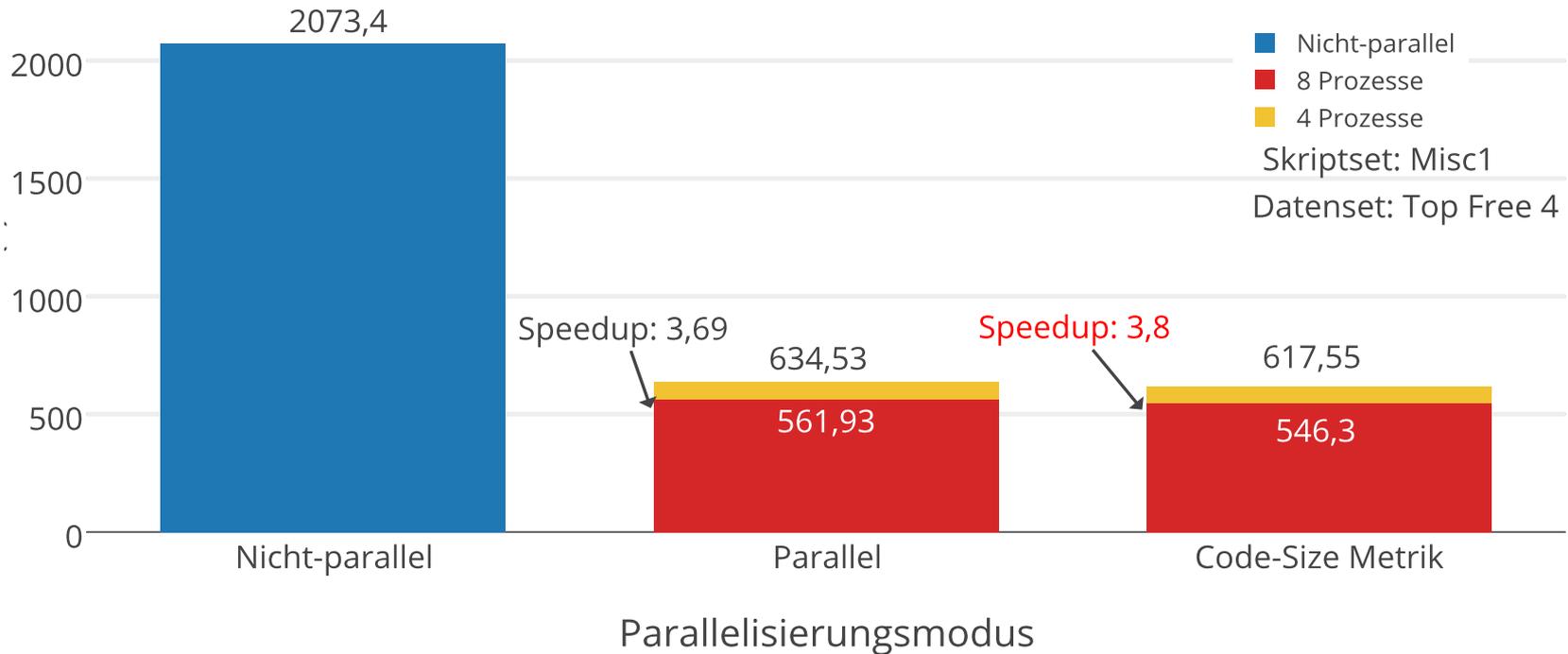
Experiments: #5

Parallele Analyse - Prozessexperiment



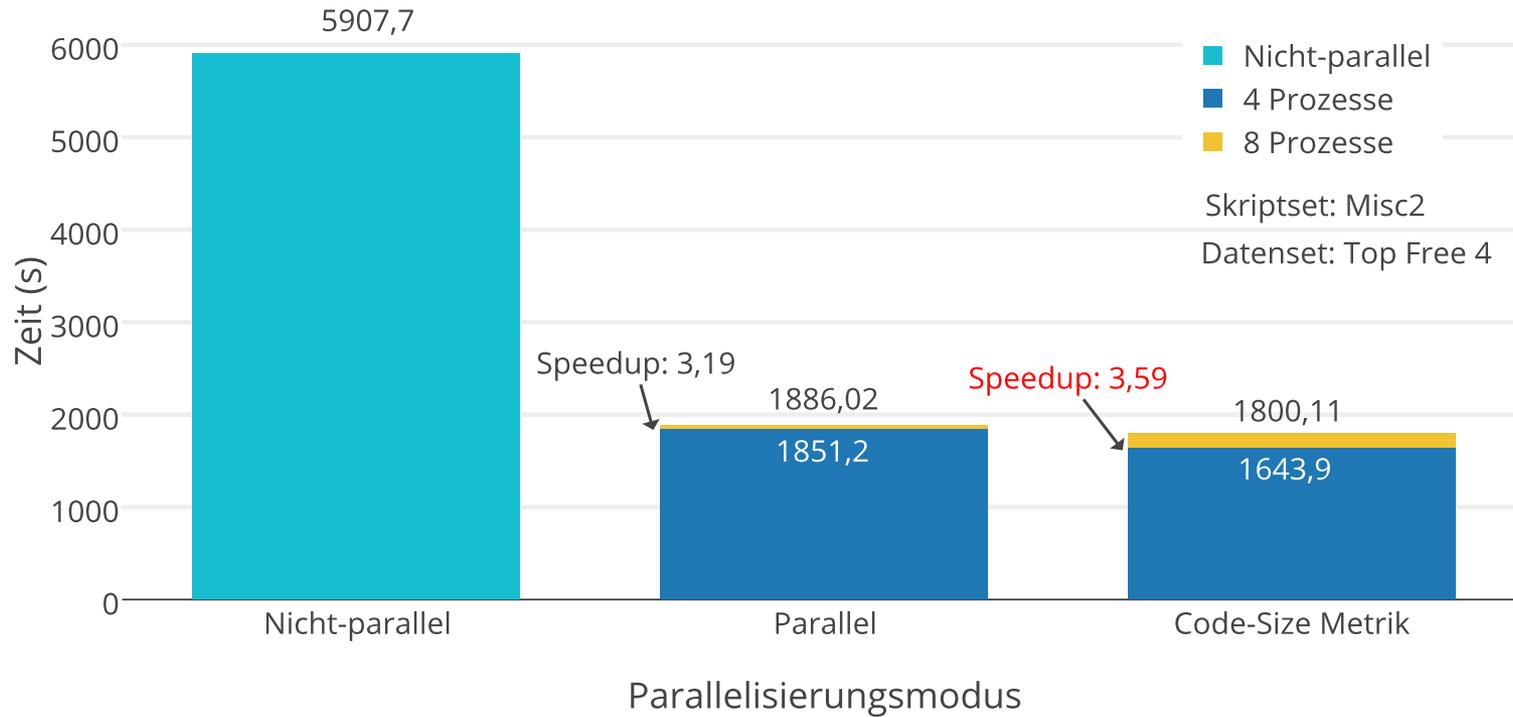
Experiments: #6

Effizienz Parallelisierung - Misc1



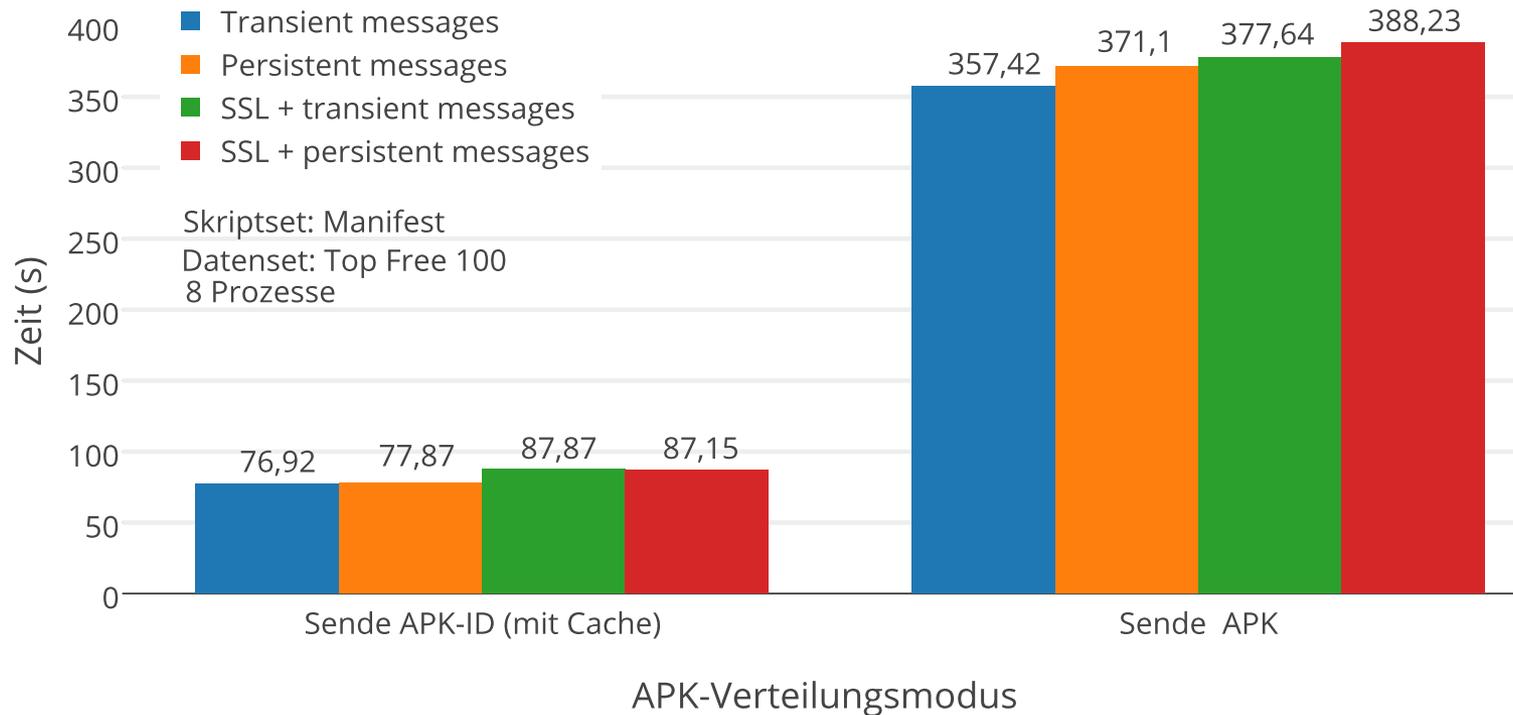
Experiments: #7

Effizienz Parallelisierung - Misc2



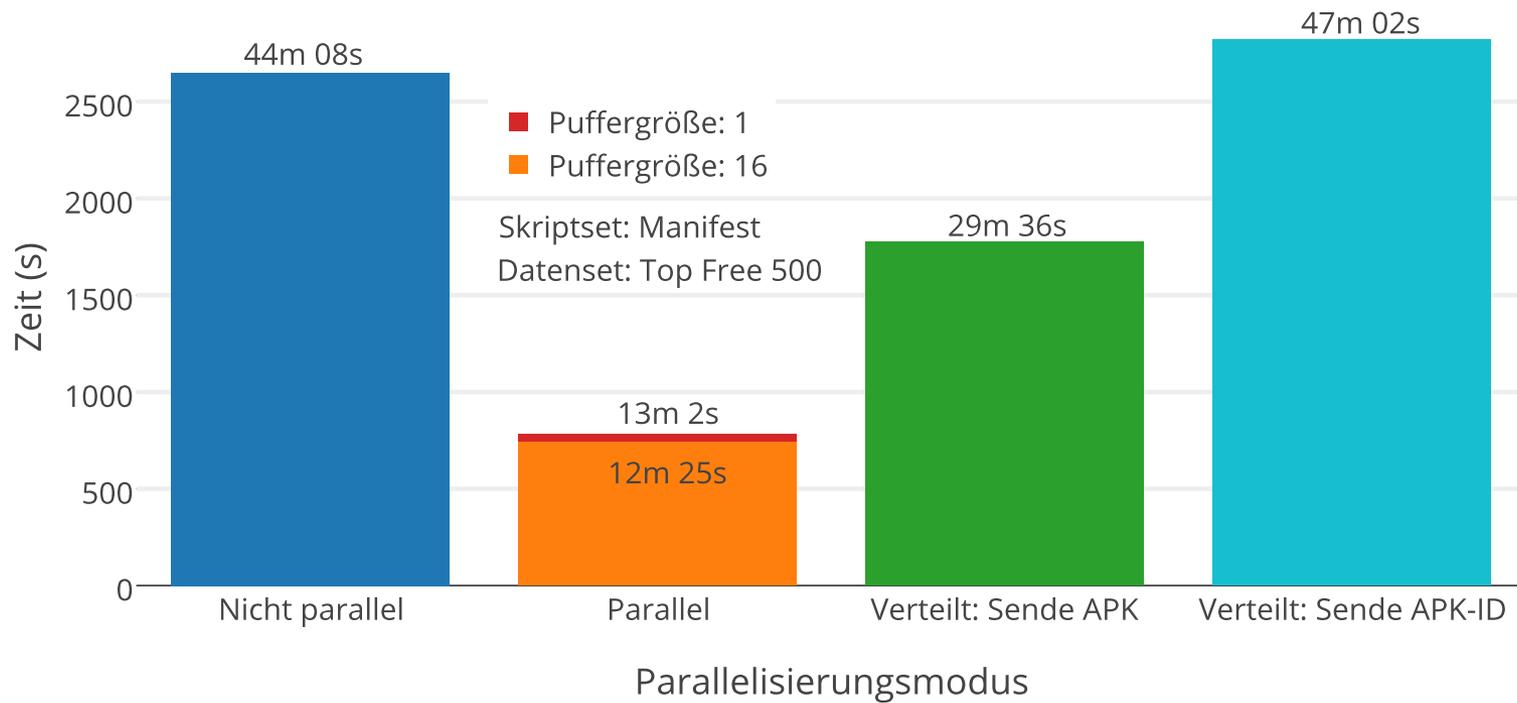
Experiments: #8

APK-Verteilung - MongoDB vs. RabbitMQ



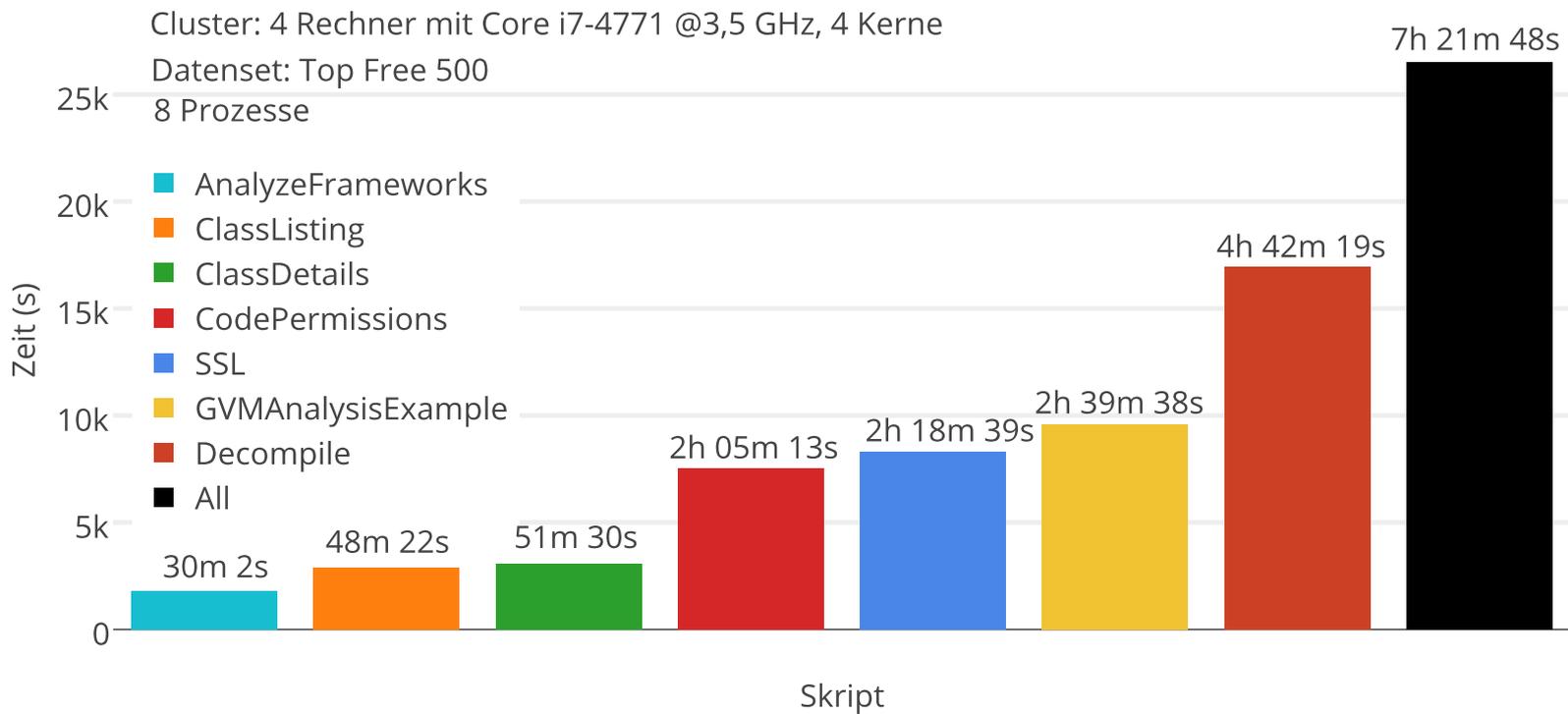
Experiments: #9

Vergleich Datendurchsatz



Experiments: #10

Analysezeiten verteilter Modus



Future work

- Analysis
 - Integrate all androguard based projects
 - AndroWarn
 - CryptoLint
 - MalloDroid
 - Decompiler cache
 - More decompilers
 - Scripts, scripts, scripts:
 - Androguard signatures for malware detection
 - Detect similarities between apps
 - Easier script-requirements
 - Add dynamic analysis functionality
 - App statistics

Future work

- Import database:
 - Regular expressions for APK filtering
- Celery:
 - More queues based on node performance
 - Routing via code size and script requirements
 - Implement remote controls -> monitoring
 - Use code-size metric for scheduling as standard
 - Use SSD for APK storage (HDD bottlenecks)

Future work

- Event-based analysis
 - Register event
 - If action triggers on next scheduled analysis
 - Inform via E-Mail
 - Example: Check if ZonerAV still vulnerable to SSL-MITM
- Message sending
 - Parallelization of message serialization
 - Integrate scripts into message → no need for script deployment anymore
- MongoDB
 - Can we further improve performance?
 - In-memory database
 - Use sharding