MCENT PERFORMANCE

March 17, 2016

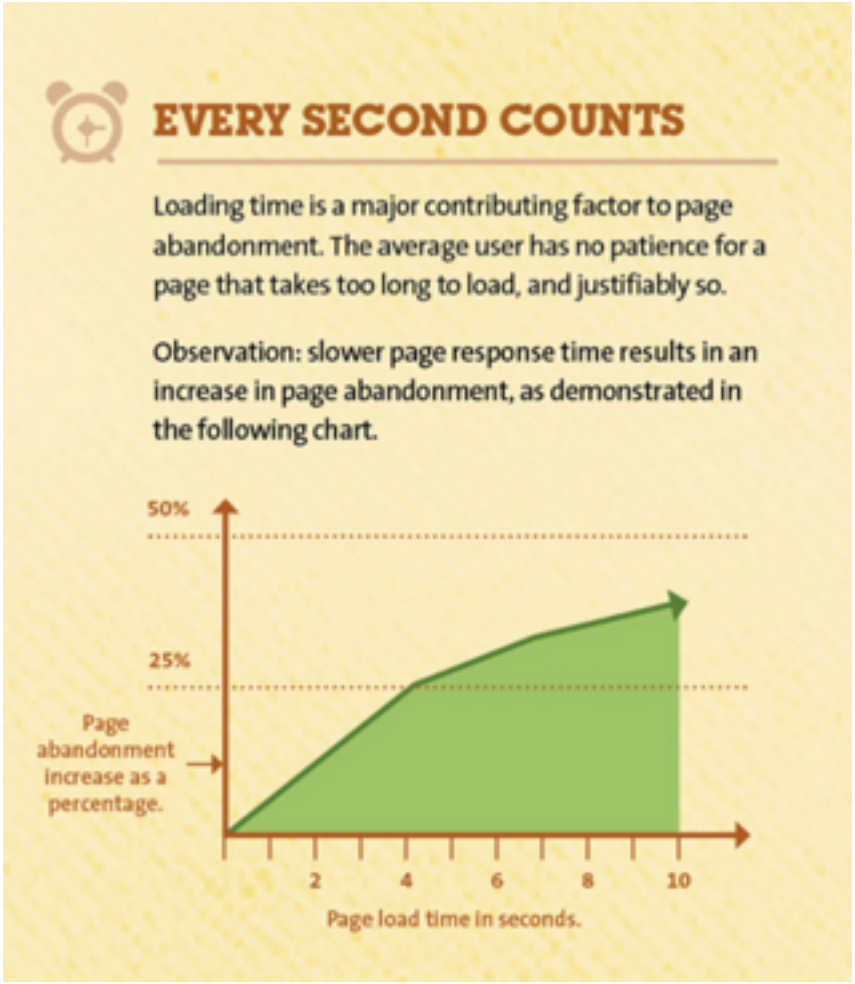
Pop quiz, hotshot.

There’s a bomb on a bus. Once the bus goes 50 miles an hour, the bomb is armed. If it drops below 50, it blows up.

What do you do? What do you do?

Speed Matters

For our business

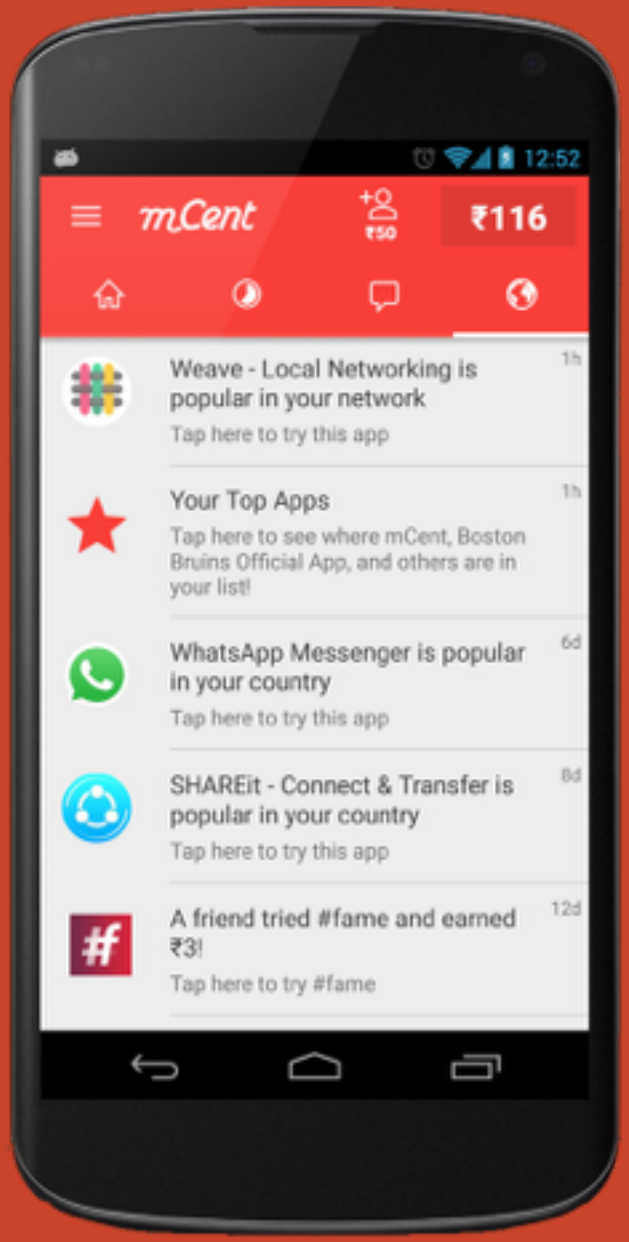


For our users

In-App Performance

mCent users are on some strange devices out there…

We have to learn how to manage the phone's resources effectively - memory, processors, etc.

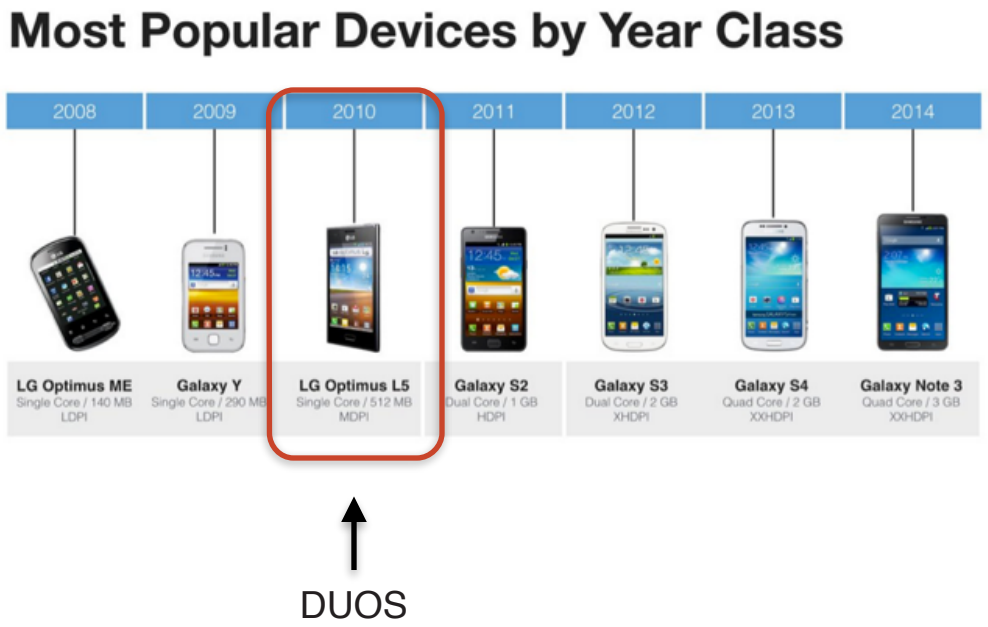


Most Popular Devices by Year Class

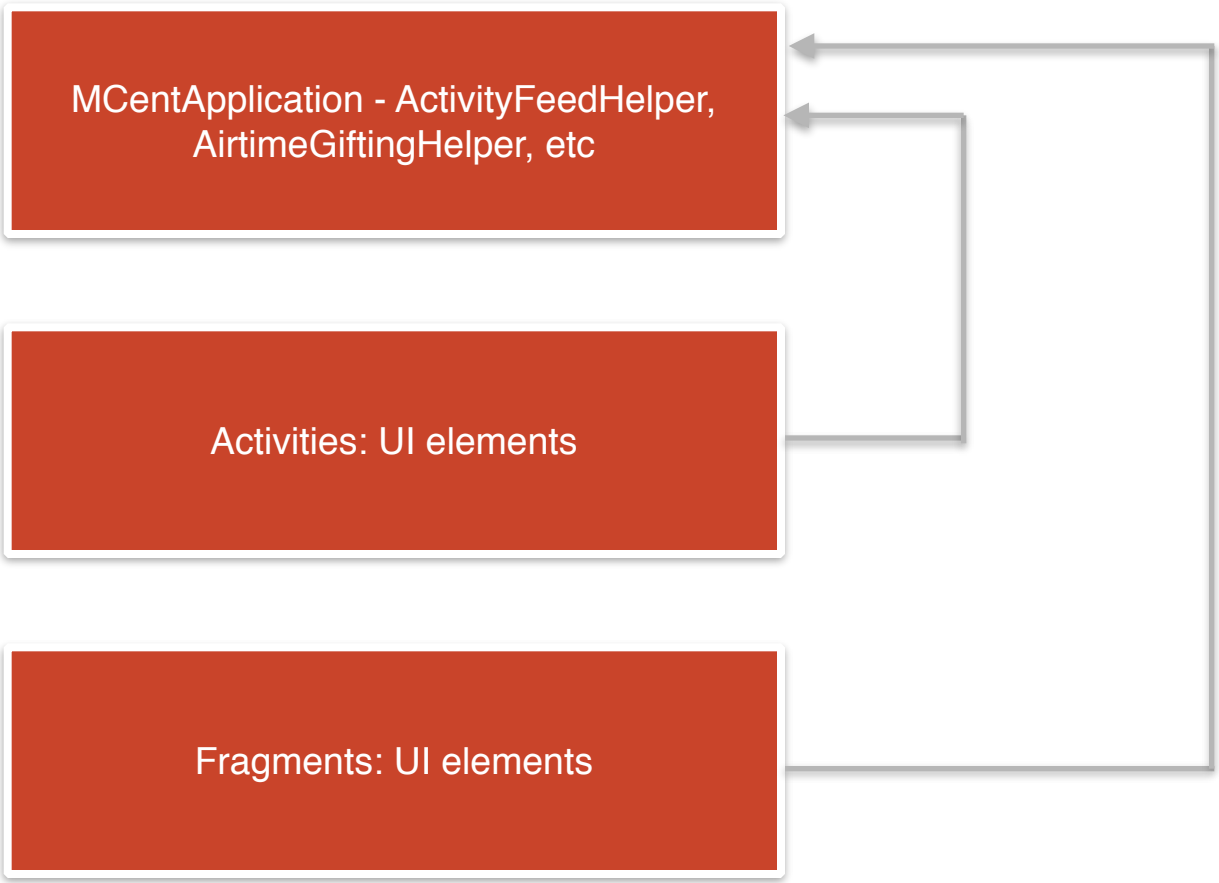
First step is understanding the hardware we're working with. What's the processor model, clock speed, number of cores we're working with, memory, etc.

For a duos:

- single core
- cortex-A5 processor
- 1GHz clock speed
- 512 MB Ram
- Android API 14/Ice Cream Sandwich



Memory Management



Background Tasks

The main thread is where all UI events must run. If we have any method modifying views, they must be run on as follows:

```
ThreadPoolUtils.startMainThreadTask(
     new Runnable() {
          @Override
          public void run() {

someView.setVisiblity(VISIBLE);
          }
});
```

Simple method to background a task so that it's running off the main thread.

```
new Thread(new Runnable() {
    @Override
    public void run() {
         doSomeLongTask();
    }
}).start();
```

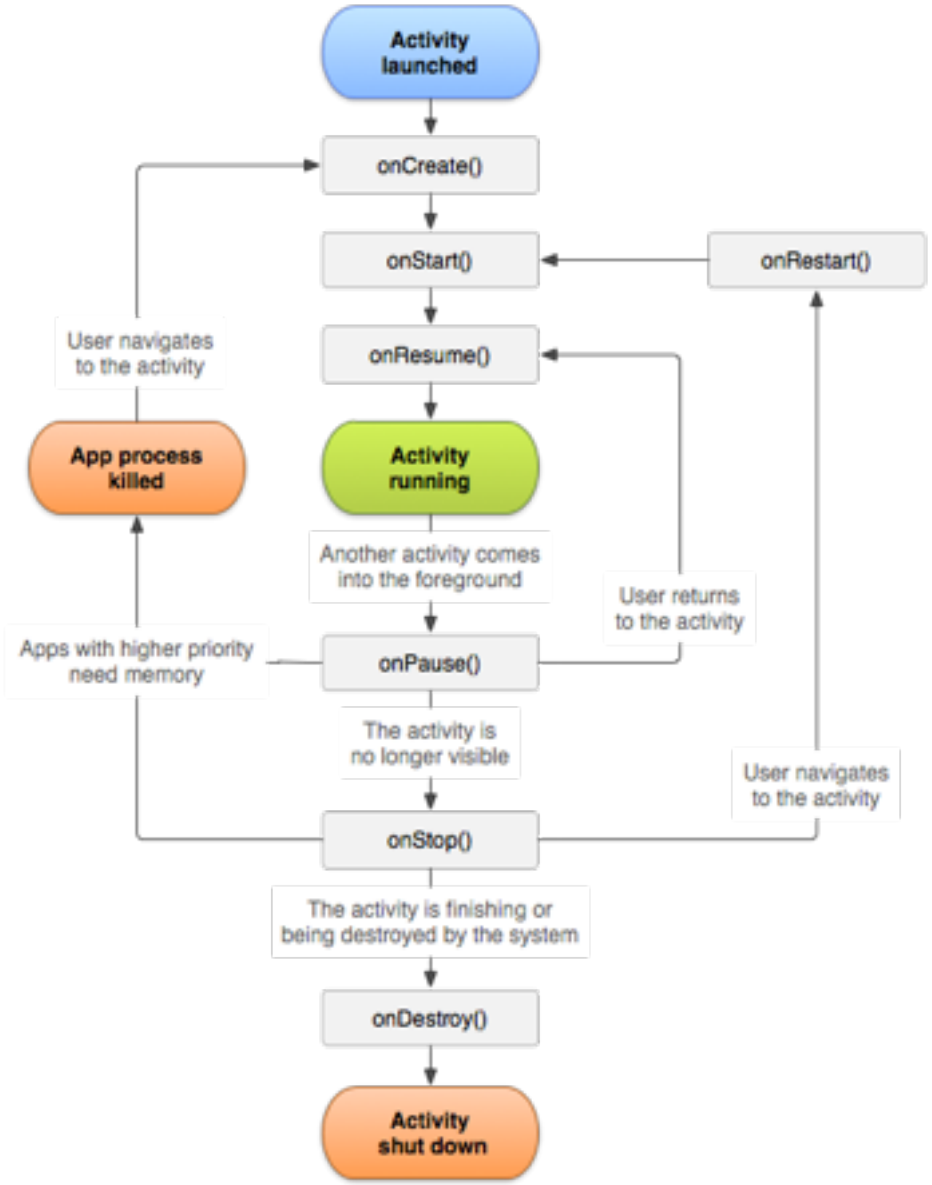
```
ThreadPoolUtils.createWorkerTask(
    WorkerTaskNames.SOME_LONG_TASK,
    new Runnable() {
        @Override
        public void run() {
            someLongTask();
        }
});
```

“If your threads don't do I/O, synchronization, etc.,and there's nothing else running, 1 thread per core will get you the best performance. However that very likely not the case. Adding more threads usually helps, but after some point, they cause some performance degradation.

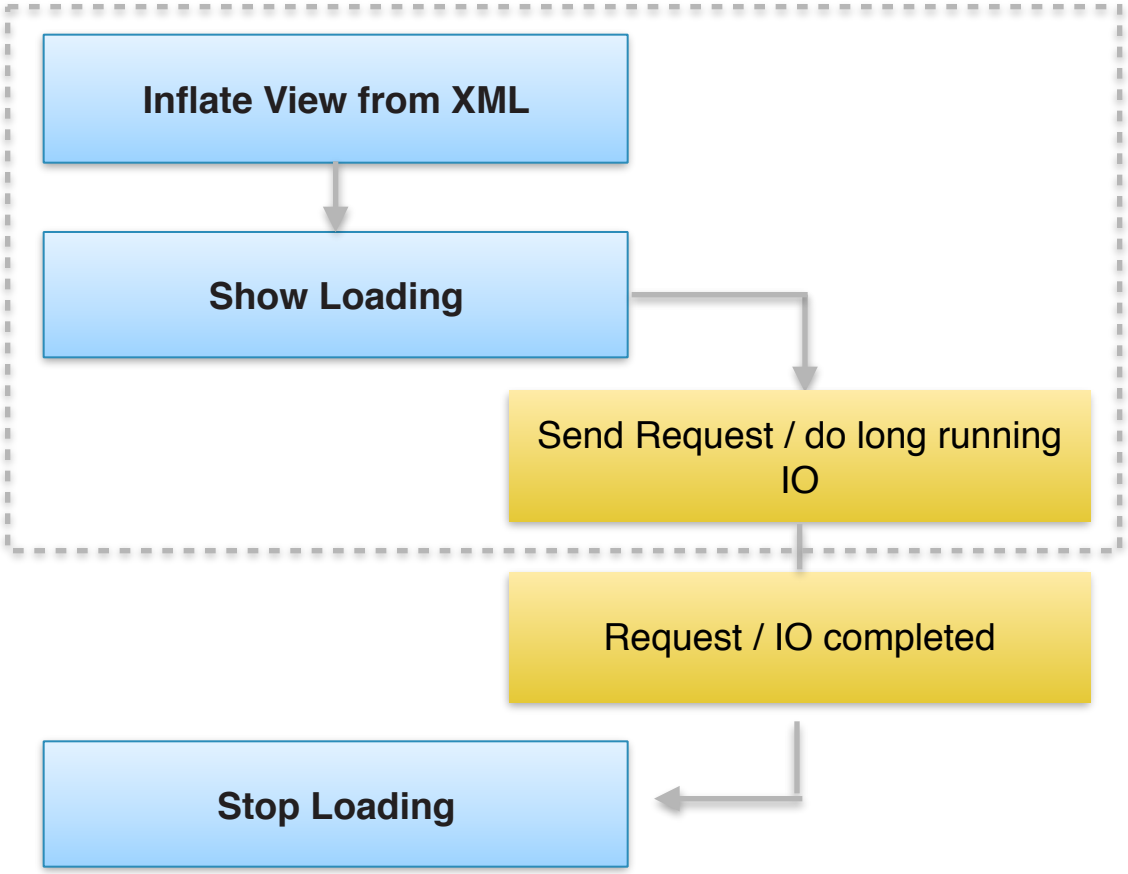
One thing for sure: 4k threads will take longer. That's a lot of context switches.”

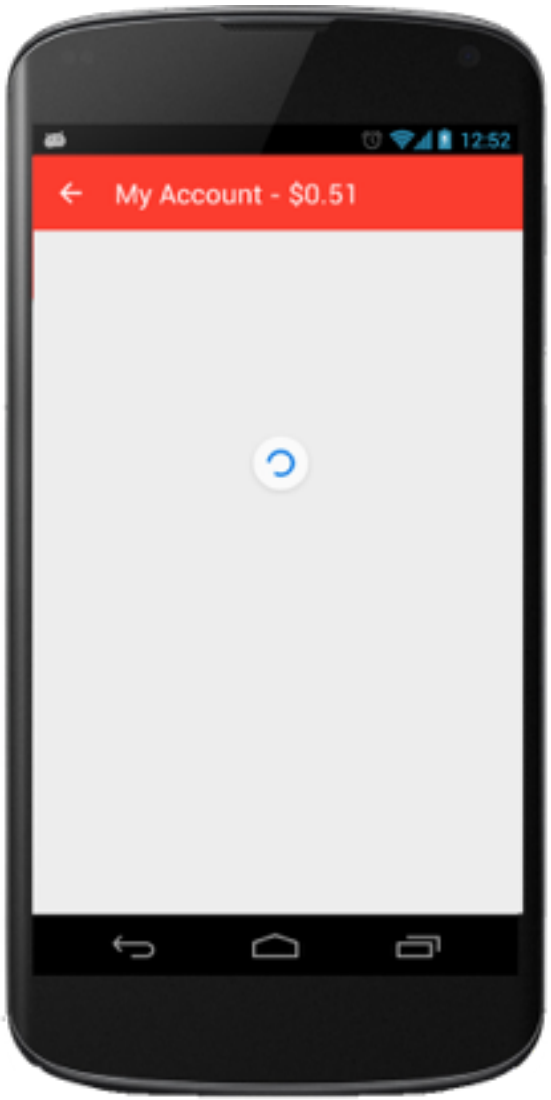
Activity Lifecycle

The important thing to note in this chart is that onCreate, onResume, onStart need to execute before the activity is considered “running”.



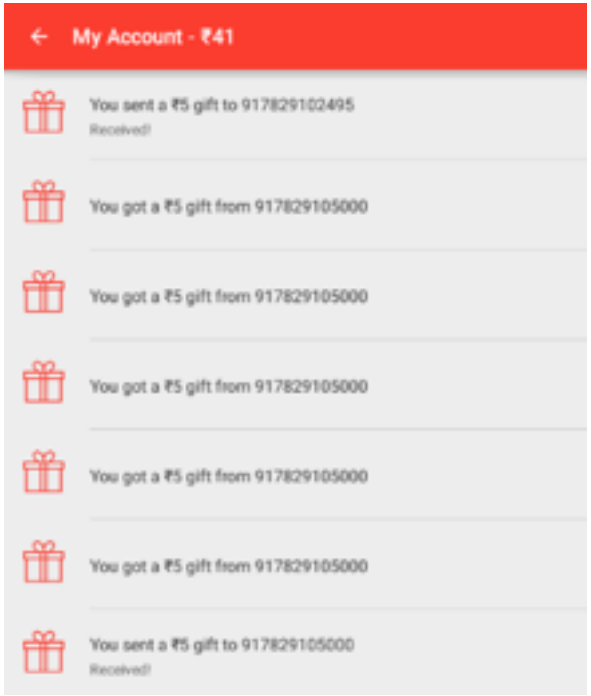
In an ideal world…





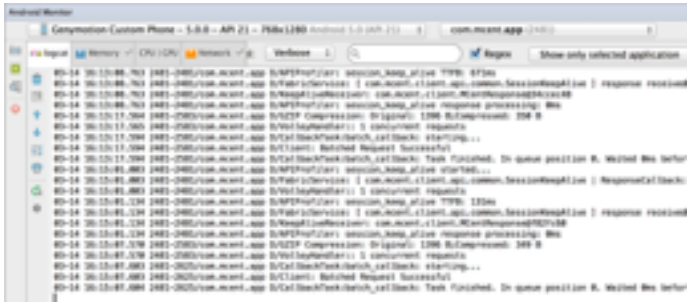
Account Activity…how can we improve this?

```
public class AccountActivity {
    ...
    protected void onCreate(Bundle savedInstanceState) {
        super.onActivityCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        ButterKnife.inject(this);

        accountHelper = mApplication.getAccountHelper();
        accountHelper.setUp(this);
        refreshActionBar();
    }

    protected void onResume(){

        /*
        Activities contain a title, description and a contact phone number.
        We must use the contact phone number to map to someone from a
        member's address book so that we can display their name on the UI.
        */
        List<Activity> accountActivities = accountHelper.getCachedAccountActivities();
        ContactDataSource contactDataSource = mApplication.getContactDataSource();
        Map<String, Contact> contactMap = contactDataSource.getAddressBookContacts()

        for(Activity activity: accountActivities){
            String phoneNumber = activity.getContactPhoneNumber();

            Contact contact = contactMap.get(phoneNumber);

            if( contact == null){
                continue;
            }

            // Displays the activity on the UI
            showActivity(contact, activity);
        }
    }
    ...
}
```



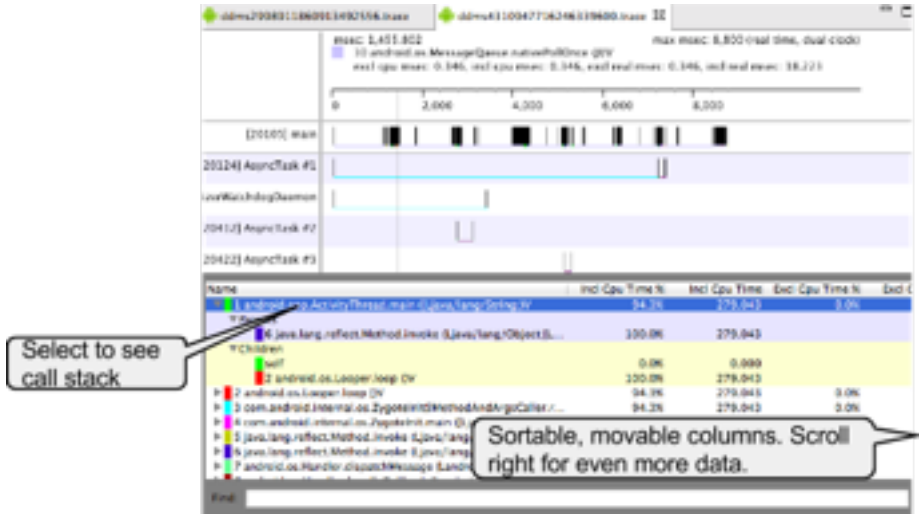
Logcat

- Filter by “PerformanceProfiler” to find timing logs of custom Kraken events
- Filter by “APIProfiler” to find information about requests
- Filter by “Task” to find information about background tasks



Profiling tools:

- Use Ross’s Systrace profiler
- Memory Monitoring / Dumps
- CPU/ GPU monitoring
- UI overdraw



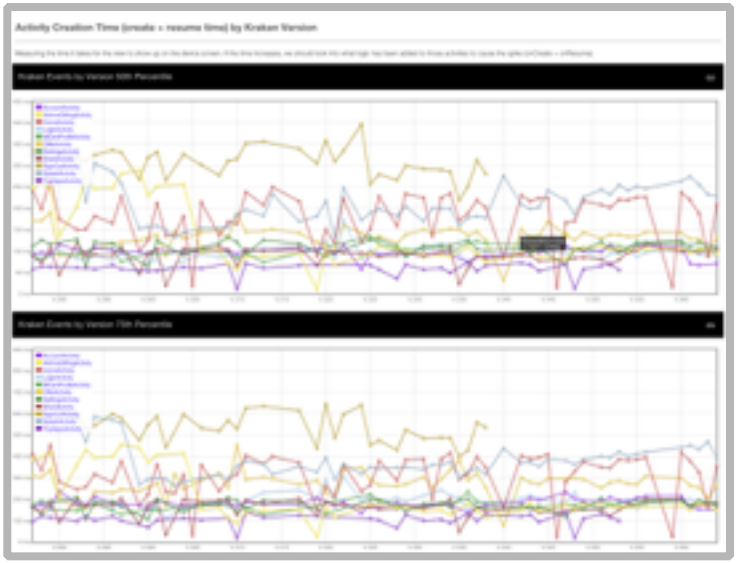
Performance Profiler
(measuring an event on production)

```
PerformanceProfiler.startKrakenTiming("doSomethingCool");
doSomethingCool();
PerformanceProfiler.stopKrakenTiming("doSomethingCool");
```

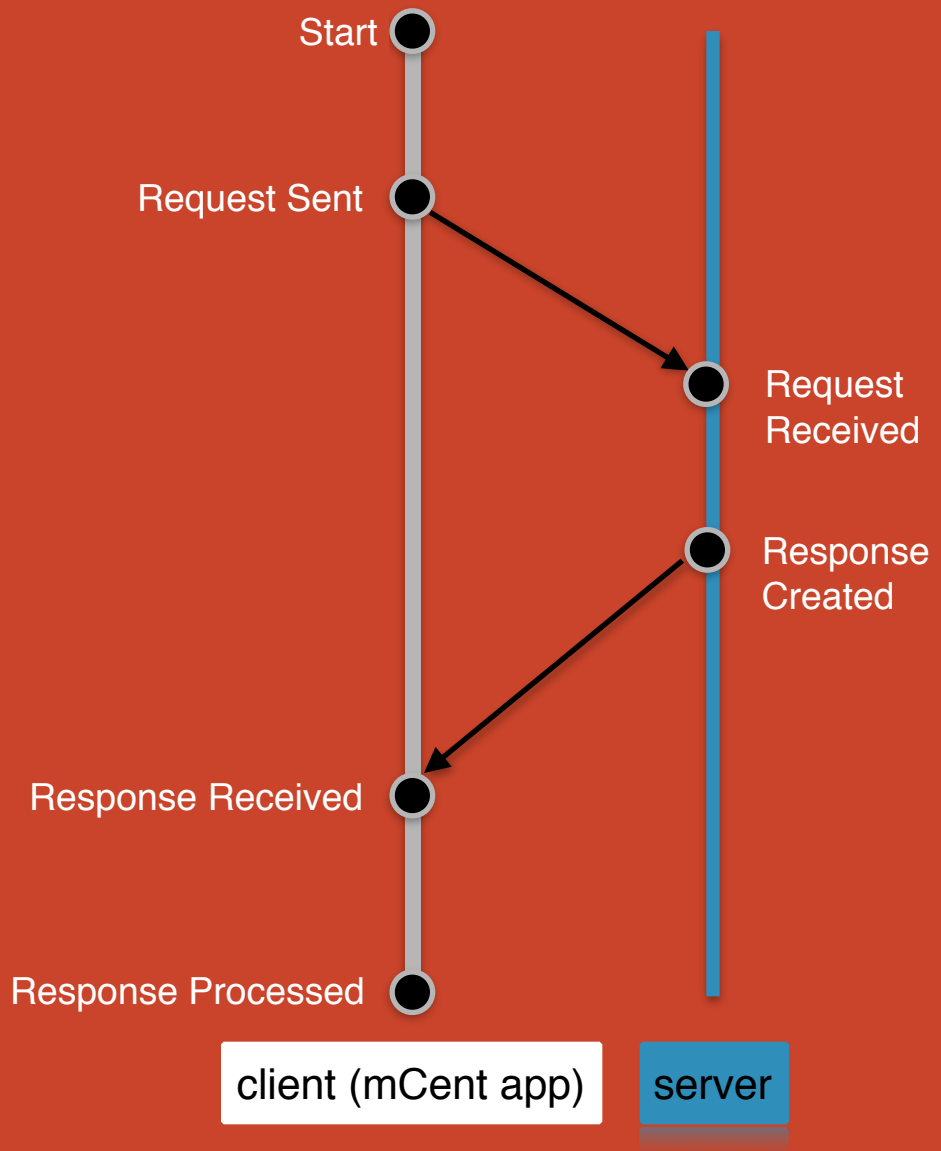
The events can be queried in snowflake:

```
SELECT
     kraken_version_num,
     timestamp_day,
     time_25,
     time_50,
     time_75,
     time_99
FROM kraken_load_time_by_version
WHERE
     kraken_version = 'kraken_version_364'
     AND timestamp_day > '2016-01-01'
     AND activity_name = 'doSomethingCool';
```

Or you can add the event to the dashboard:



Network Performance



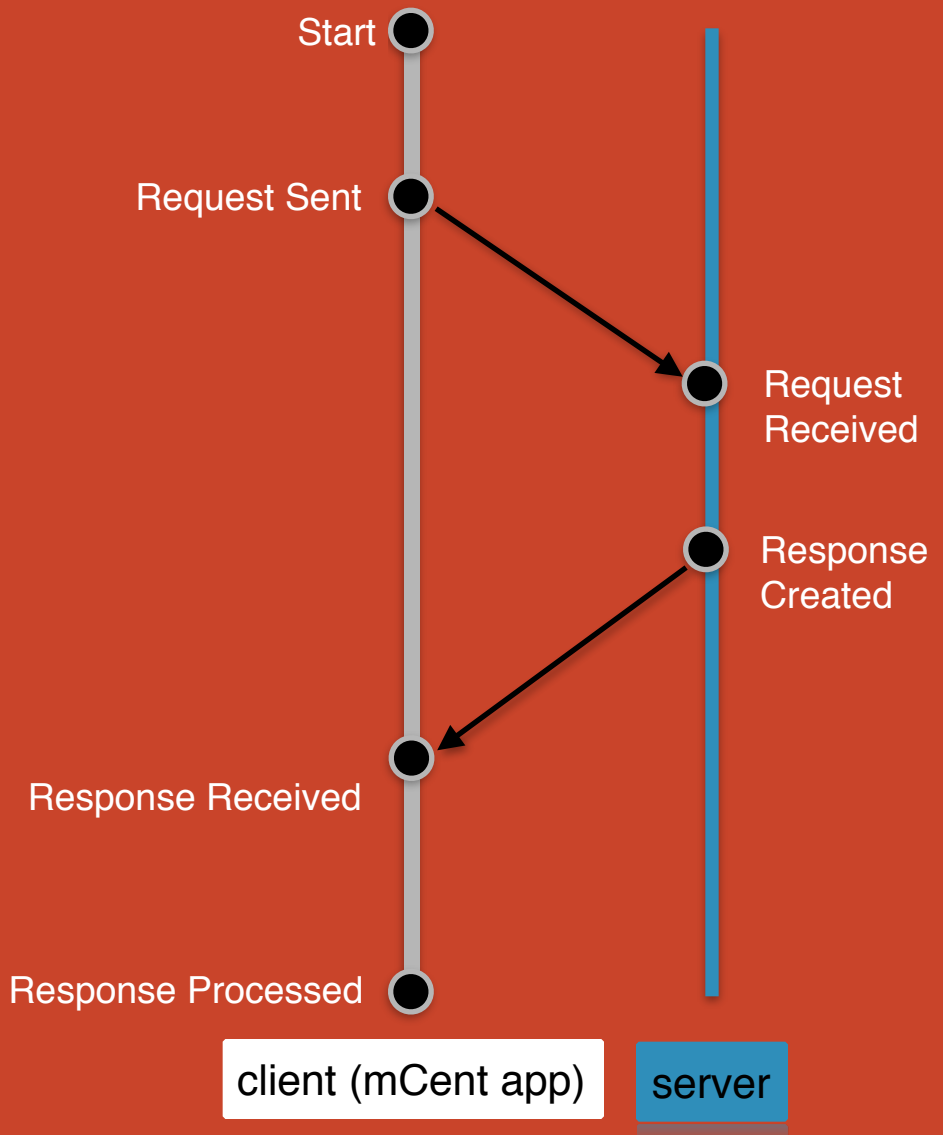
Network Performance

Preprocess Time:
The time from when a request is initialized to when it was sent.

Server Processing Time:
The time when the request is received to when the response is created (ie. how long it takes the endpoint to execute)

Response Processing Time / Postprocess:
Time from when the response is received to when it is finished processing on the client. (can contain UI elements)

Time to First Byte (TTFB):
The time from when the request is started until when the response is received.



Let's say we have concurrent requests from Kraken that hit these 2 endpoints.

```python
@api.route('/v1/get_offers')
@lock_pcid()
def get_offers():
    params = request.json
    kraken_version = params.get("kraken_version")
    session_id = (
        params.get('auth_token') or
        params.get("session_id")
    )

    session_member = session_manager.get_session_member(
        session_id
    )
    offers = offer_manager.get_offers(
        session_member
    )

    info = kraken_configurations.get_info_for_member(
        session_member
    )

    return jsonify({
        'configuration': info,
        'offers': offers
    })
```

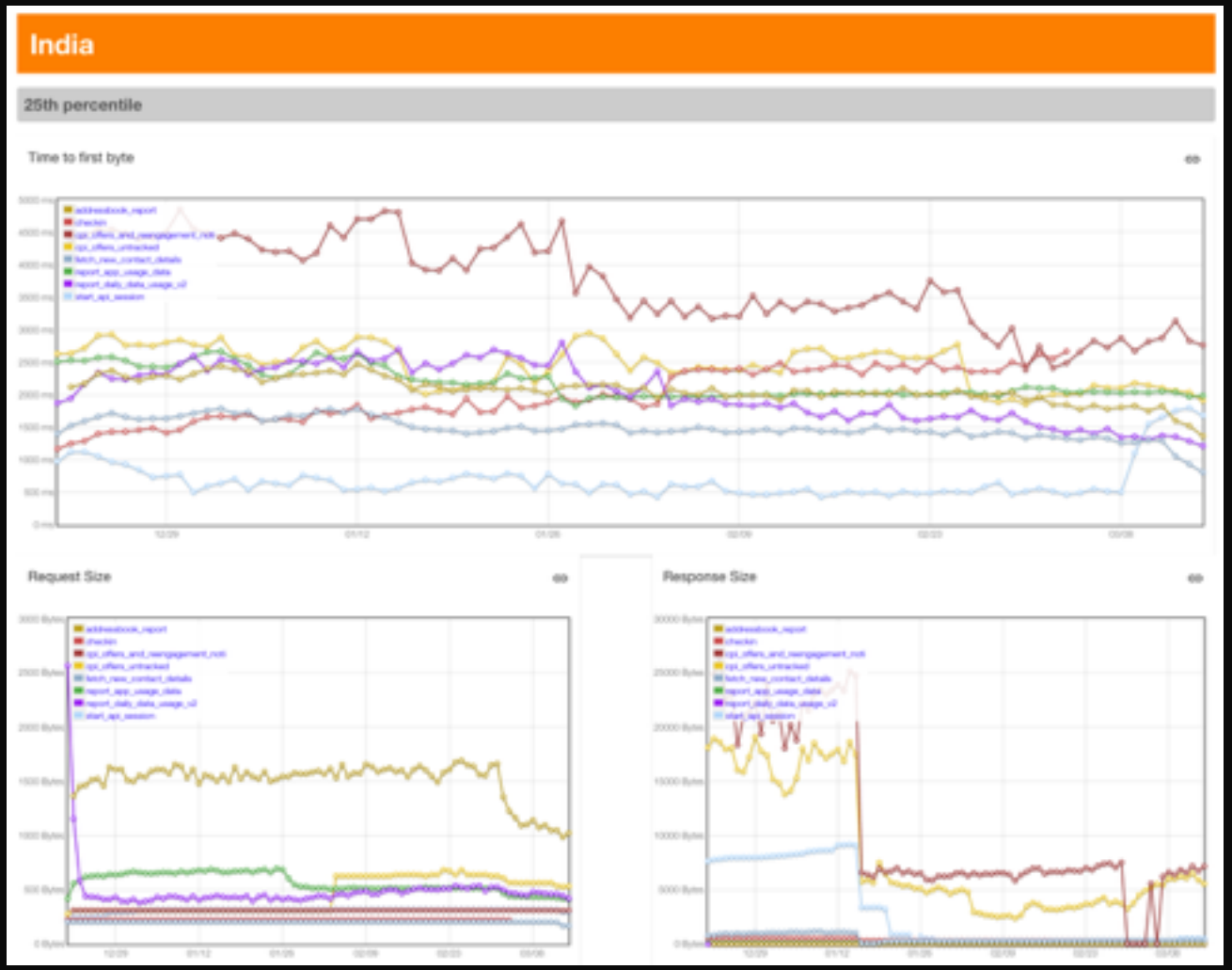
```python
@api.route('/v1/get_balance', methods=['POST'])
@lock_pcid()
def get_balance():
    params = request.json
    session_id = (
        params.get('auth_token') or
        params.get("session_id")
    )
    result = {
        "balance": (
            current_app.managers.mcent_profile.get_balance(
                session_id
            )
        )
    }
    return jsonify(result)
```

What's wrong with this?

Slows down server-side processing

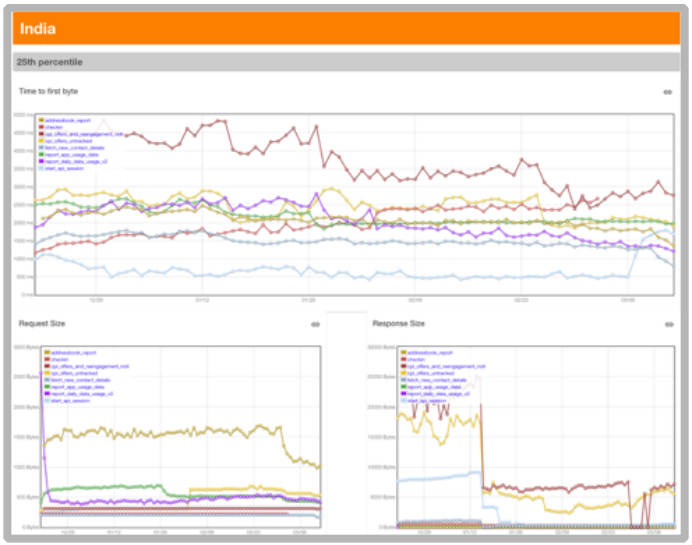
```
@lock_pcid()
    params = request.json
    kraken_version = params.get("kraken_version")
    session_id = (
        params.get('auth_token') or
        params.get("session_id")
    )

    session_member = session_manager.get_session_member(
        session_id
    )
    offers = offer_manager.get_offers(
        session_member
    )

    info = kraken_configurations.get_info_for_member(
        session_member
    )

    return jsonify({
        'configuration': info,
        'offers': offers
    })
```

Monitoring Performance



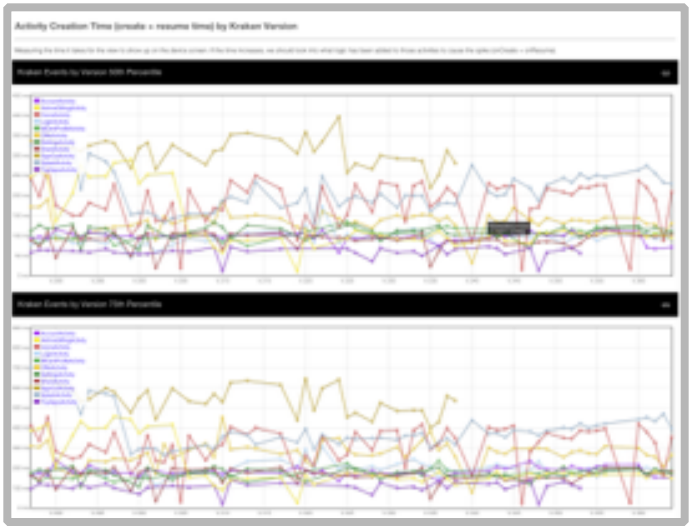
API Graphs (Dashboard)

Monitors the performance of each endpoint by country. Tracks time to first byte (TTFB), response processing, request/response size, and server processing time.



Kraken Events (Dashboard)

Monitors the performance of custom, in-app events. Focuses on the activity lifecycle and tracks how long those activities take to load.



Mobile Network (Dashboard)

Monitors country-level trends to better understand how bandwidth is changing. Note, this gives a us an approximation because we don't take extremely large files for our users to download to give a better sense of max bandwidth.

Experiment Analysis (Dashboard)

Any active experiment has analytics that evaluate business metrics. This tool compares control (variant 0) vs other variants in the experiment and points out changes in app events and network timings. In the experiment manager, select an experiment and then click 'analyze' at the top. Scroll below the graphs to see the performance impact.

Performance Improvements

	LOW IMPACT	HIGH IMPACT
LITTLE EFFORT	• *SQLITE OPTIMIZATION ON ANDROID* • *KRAKEN CONFIGURATION*	• *GZIP COMPRESSION* • *BACKGROUND TASKS* • *REQUEST PRIORITY* • *ACTIVITY START OPTIMIZATION*
LARGE EFFORT	• *ROUTE REQUESTS THROUGH PROXY SERVERS*	• *BATCH CALLS* • *CACHING* • *BACKGROUND CALLBACKS* • *GARBAGE COLLECTION*