Sunfund Investment & Management Group Co., Ltd

A brief talk about Android performance

Huqiu Liao

Agenda

Overview

- Aspects
- Tools
- Problems: Trade-offs vs rules

Principles and examples

- Principles
- Examples

Challenges and solutions

- Challenges
- Solutions

Overview

Android performance overview

1. Overview

1. CPU
2. GPU
3. Network / Data
4. Memory
5. Battery
6. ...

2. Tools

1. Android Studio: Memory / GPU / CPU ...
2. Battery historian
3. Leakcanary
4. Infer
5. Performance Log & BI
6. ...

What’s the problem?

Android performance overview

1. Overview

1. CPU
2. GPU
3. Network / Data
4. Memory
5. Battery
6. ...

2. Tools

1. Android Studio: Memory / GPU / CPU ...
2. Battery historian
3. Leakcanary
4. Infer
5. Performance Log & BI
6. ...

3. Problems

1. A lot of trade-offs
2. Depended on scenarios

Principles and examples

A brief talk about Android performance

Reduce calculation

AOT calculation

Defer calculation

Reduce calculation

Reduce calculation

Time Complexity

Time vs Space

Cache

Only do necessary work

Consider time complexity

Reduce calculation AOT calculation Defer calculation

Store child view in SparseArray in RecyclerView.ViewHolder

```
42
43      /**
44       * https://github.com/CymChad/BaseRecyclerViewAdapterHelper
45       */
46      public class BaseViewHolder extends RecyclerView.ViewHolder {
47
48          /**
49           * Views indexed with their IDs
50           */
51          private final SparseArray<View> views;
52
53          private final LinkedHashSet<Integer> childClickViewIds;
54          private final LinkedHashSet<Integer> itemChildLongClickViewIds;
55
```

Consider time complexity

So that it can save some code to access these children

```
public class QuickAdapter extends BaseQuickAdapter<Status, BaseViewHolder> {
    public QuickAdapter() {
        super(R.layout.tweet, DataServer.getSampleData());
    }

    @Override
    protected void convert(BaseViewHolder viewHolder, Status item) {
        viewHolder.setText(R.id.tweetName, item.getUserName())
                .setText(R.id.tweetText, item.getText())
                .setText(R.id.tweetDate, item.getCreatedAt())
                .setVisible(R.id.tweetRT, item.isRetweet())
                .linkify(R.id.tweetText);
                 Glide.with(mContext).load(item.getUserAvatar()).crossFade().into((
    }
}
```

What’s the time complexity?

Consider time complexity

Reduce calculation AOT calculation Defer calculation

O(lgn)

```
42
43      /**
44       * https://github.com/CymChad/BaseRecyclerViewAdapterHelper
45       */
46      public class BaseViewHolder extends RecyclerView.ViewHolder {
47
48          /**
49           * Views indexed with their IDs
50           */
51          private final SparseArray<View> views;
52
53          private final LinkedHashSet<Integer> childClickViewIds;
54          private final LinkedHashSet<Integer> itemChildLongClickViewIds;
55
```

Consider time complexity

O(nlogn) !

```
public class QuickAdapter extends BaseQuickAdapter<Status, BaseViewHolder> {
    public QuickAdapter() {
        super(R.layout.tweet, DataServer.getSampleData());
    }

    @Override
    protected void convert(BaseViewHolder viewHolder, Status item) {
        viewHolder.setText(R.id.tweetName, item.getUserName())
                .setText(R.id.tweetText, item.getText())
                .setText(R.id.tweetDate, item.getCreatedAt())
                .setVisible(R.id.tweetRT, item.isRetweet())
                .linkify(R.id.tweetText);
                 Glide.with(mContext).load(item.getUserAvatar()).crossFade().into((
    }
}
```

Time and space trade-offs

Calculation is expensive

Store result in memory or disk.

IO is expensive

Cache data in memory / file cache

Cache

Docker

Strategy / scenario

What kind of data should put into cache?

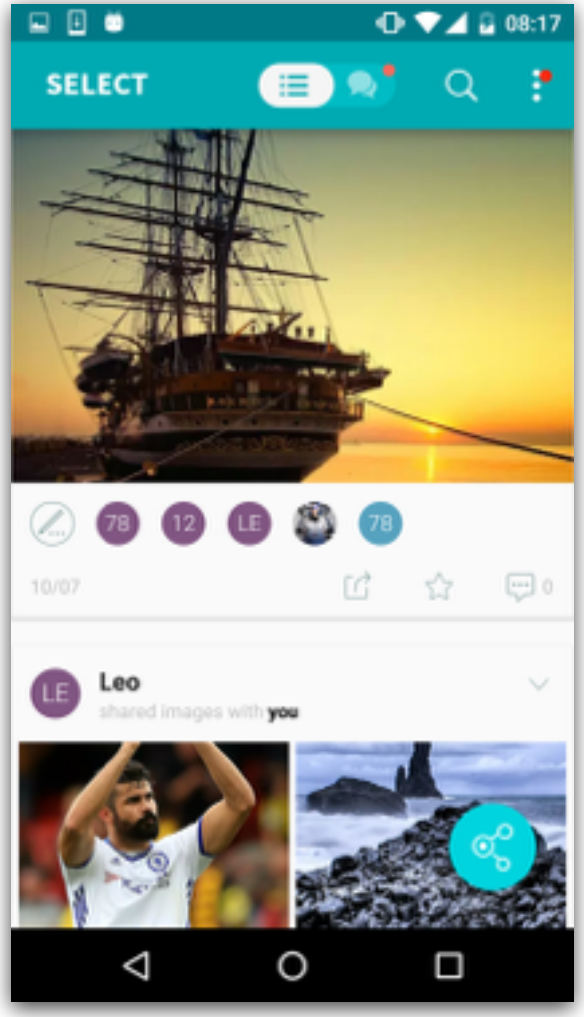
when to remove them?

LruCache

Size

Cache is not free

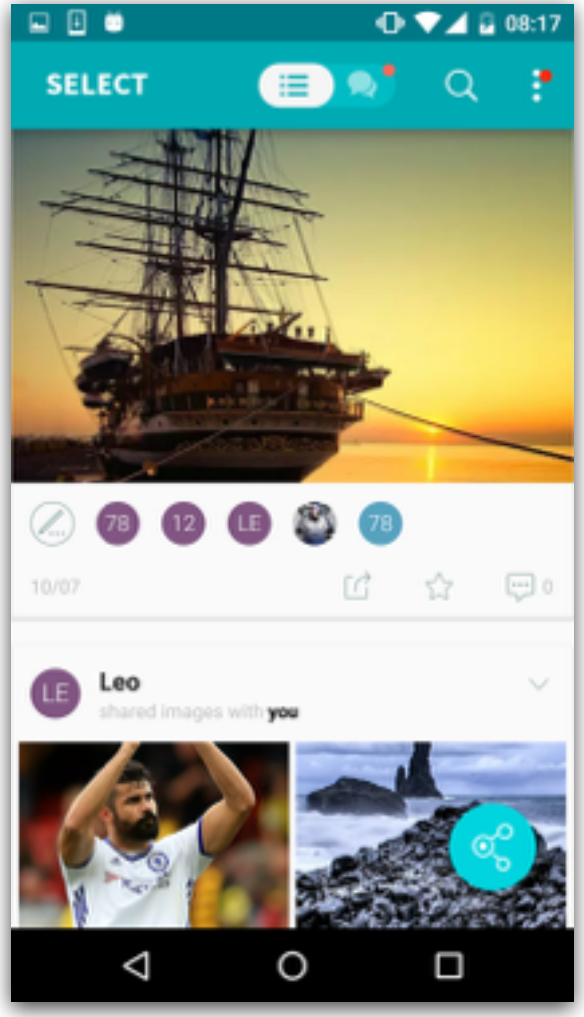
Cache: RecycledViewPool



Using RecycledViewPool to cache scape view

```
33
34      public static void bindViewHolderForFooterView(CubeRecyclerViewAdapter<?>
35          adapter.setViewHolderClass(
36                  FeedListItemFooterItem.VIEW_TYPE_EDIT, null,
37                  FeedListItemFooterItemEditViewHolder.class, 16);
38          adapter.setViewHolderClass(
39                  FeedListItemFooterItem.VIEW_TYPE_USER, null,
40                  FeedListItemFooterItemUserViewHolder.class, 48);
41      }
```

Cache: RecycledViewPool



Using RecycledViewPool to cache scape view

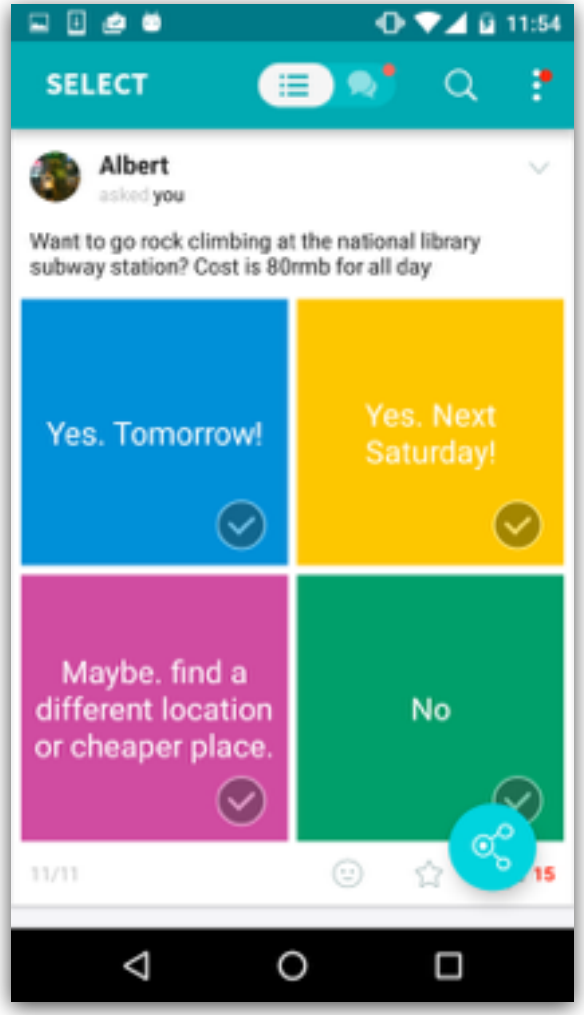
```
240        private void init() {
241            View view = LayoutInflater.from(getContext()).inflate(R.layout.feed_list_item_footer,
242
243            mRecyclerView = (RecyclerView) view.findViewById(R.id.recepient_list);
244            LinearLayoutManager layoutManager = new LinearLayoutManager(view.getContext());
245            layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
246            mRecyclerView.setOnScrollListener(sOnScrollListener);
247            mRecyclerView.setHasFixedSize(true);
248            mRecyclerView.setLayoutManager(layoutManager);
249            mRecyclerView.setItemAnimator(null);
250
251            final RecyclerView.RecycledViewPool userViewPool = FeedListDataModel.getInstance()
252                    .getFeedListItemViewHolderManager().getRecycledViewPoolForFooterView();
253
254            mRecyclerView.setRecycledViewPool(userViewPool);
255            mRecyclerView.setAdapter(mRecyclerViewAdapter);
256
257            FeedListItemViewHolderManager.bindViewHolderForFooterView(mRecyclerViewAdapter);
```

Only do necessary work

Only update the view needs to be updated

```
592     private void startExpiration(final ContentModel content, final boolean mine) {
593         expirationView.setVisibility(View.VISIBLE);
594
595         if (content.isExpired()) {
596             stopExpiration();
597         } else {
598             expirationView.setText(getExpirationString(content));
599             expirationHandler.postDelayed(new Runnable() {
600                 @Override
601                 public void run() {
602                     startExpiration(content, mine);
603                 }
604             }, 1000);
605         }
606     }
607
608     private String getExpirationString(ContentModel content) {
609         long time = content.getUpdateAt() + content.getExpireAt() - System.currentTimeMillis();
610         if (time < 1000) {
611             return "";
612         }
613         return TimeUtil.formatExpirationTime(time);
614     }
```

Only do necessary work



Avoid creating unnecessary objects

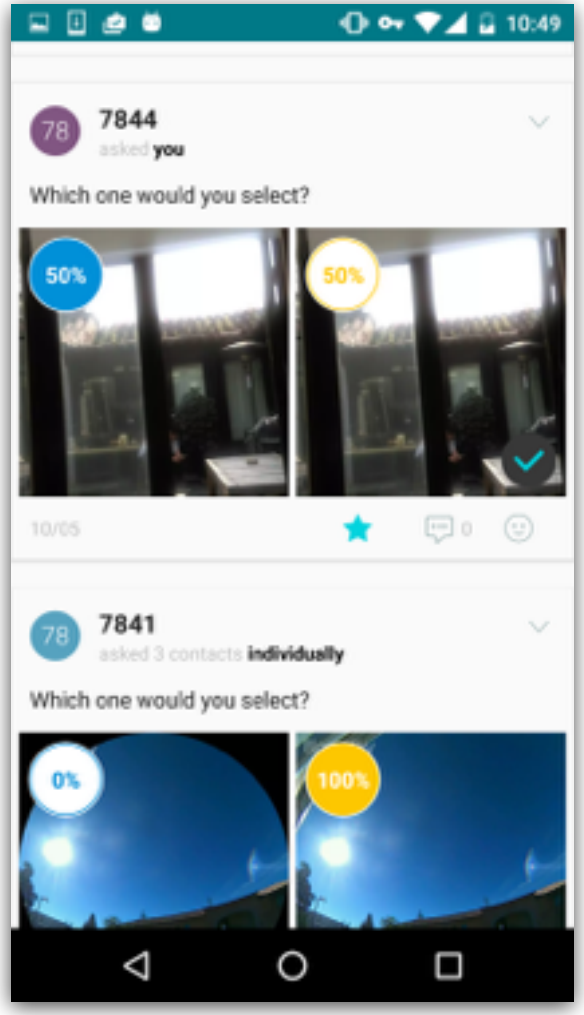
```
46          if (!mine) {
47              //no any select
48              if (selection == PollPost.SELECTION_NONE) {
49                  leftButton.setEnabled(true);
50                  leftButton.setOnClickListener(new View.OnClickListener() {
51                      @Override
52                      public void onClick(View view) {
53                          selectOption(poll, PollPost.SELECTION_YES);
54                      }
55                  });
56                  rightButton.setEnabled(true);
57                  rightButton.setOnClickListener(new View.OnClickListener() {
58                      @Override
59                      public void onClick(View view) {
60                          selectOption(poll, PollPost.SELECTION_NO);
61                      }
62                  });
63              //have select
64              } else {
65                  leftButton.setChecked(selection == PollPost.SELECTION_YES);
66                  rightButton.setChecked(selection == PollPost.SELECTION_NO);
67              }
68          }
```

“

It’s not necessary to create it every time.

Calculate ahead of time

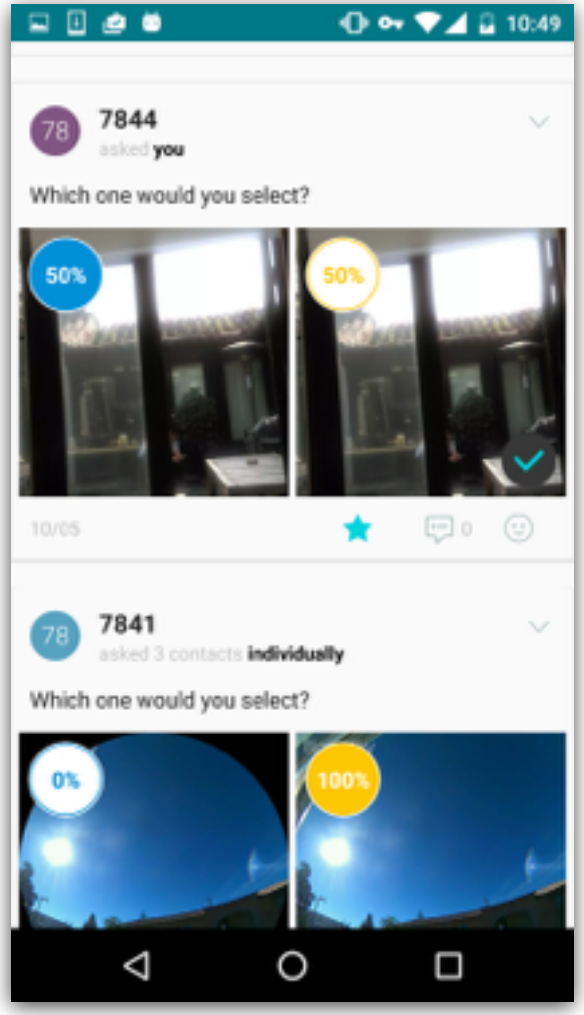
Calculate when scrolling the list



Build SpannableString and Html

```
70          if (mine || selection != PollPost.SELECTION_NONE) {
71              int totalVotes = poll.getNumOfVotes();
72              int yesVotes = poll.getNumOfVotes(PollPost.SELECTION_YES);
73              int noVotes = poll.getNumOfVotes(PollPost.SELECTION_NO);
74
75              int yesPercent = totalVotes != 0 ? (int)((double)yesVotes / (double)totalVote
76              int noPercent = totalVotes != 0 ? (int)((double)noVotes / (double)totalVotes
77
78              leftRating.setText(yesPercent + "%");
79              leftRating.setTextColor(this.view.getResources().getColor(yesVotes > noVotes
80              leftRating.setBackgroundResource(yesVotes > noVotes ? R.drawable.bg_poll_winn
81              leftRating.setVisibility(View.VISIBLE);
82
83              rightRating.setText(noPercent + "%");
84              rightRating.setTextColor(this.view.getResources().getColor(noVotes > yesVotes
85              rightRating.setBackgroundResource(noVotes > yesVotes ? R.drawable.bg_poll_win
86              rightRating.setVisibility(View.VISIBLE);
87          } else {
88              leftRating.setVisibility(View.GONE);
89              rightRating.setVisibility(View.GONE);
90          }
```

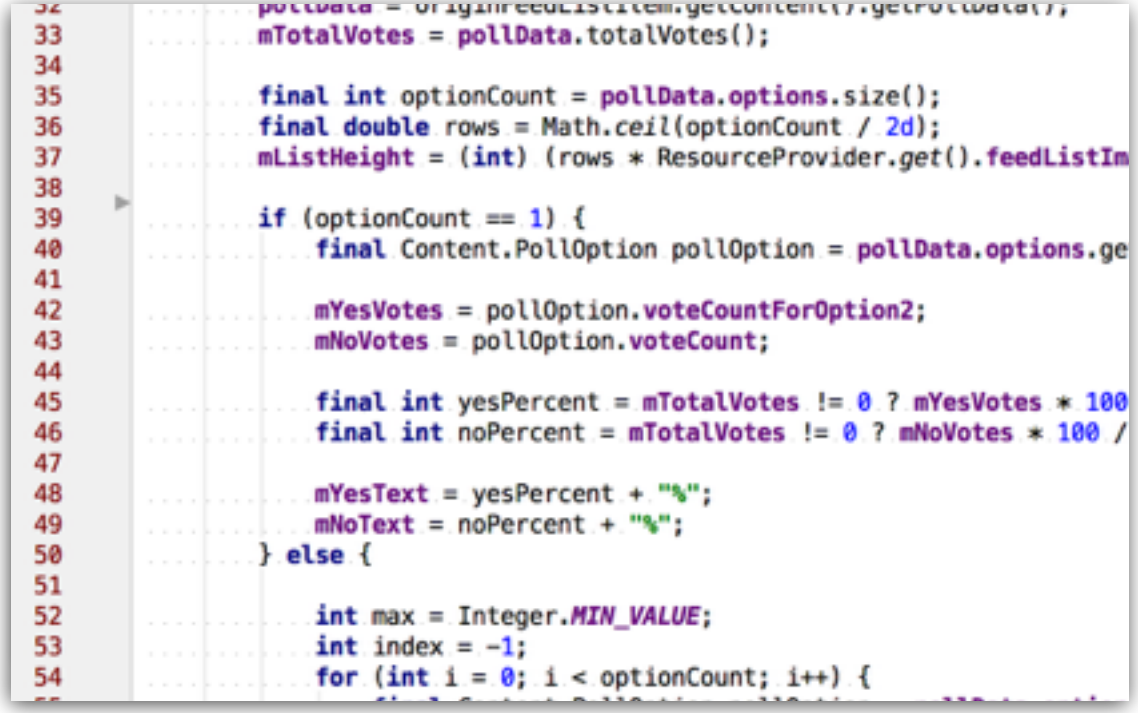
Calculate when scrolling the list



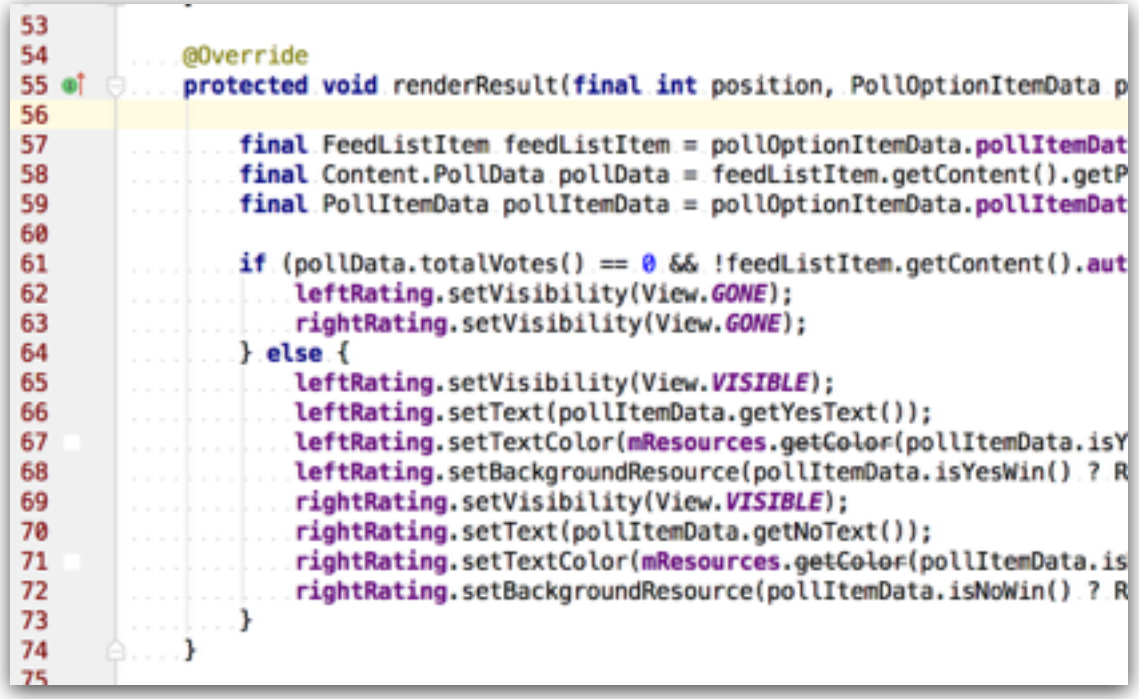
Build Fresco DraweeController

```
60
61      public static void loadImage(SESimpleDraweeView simpleDraweeView,
62                                   @Nullable SEImageLoadParam loadParam) {
63          if (loadParam == null) {
64              simpleDraweeView.setController(null);
65              return;
66          }
67          String url = loadParam.getUrl();
68          DraweeController draweeController = Fresco.newDraweeControllerBuilder()
69                  .setAutoPlayAnimations(true)
70                  .setUri(url)
71                  .build();
72          simpleDraweeView.setController(draweeController);
73      }
```

DataModel & View: SpannableString



```
33      mTotalVotes = pollData.totalVotes();
34
35      final int optionCount = pollData.options.size();
36      final double rows = Math.ceil(optionCount / 2d);
37      mListHeight = (int) (rows * ResourceProvider.get().feedListIm
38
39      if (optionCount == 1) {
40          final Content.PollOption pollOption = pollData.options.ge
41
42          mYesVotes = pollOption.voteCountForOption2;
43          mNoVotes = pollOption.voteCount;
44
45          final int yesPercent = mTotalVotes != 0 ? mYesVotes * 100
46          final int noPercent = mTotalVotes != 0 ? mNoVotes * 100 /
47
48          mYesText = yesPercent + "%";
49          mNoText = noPercent + "%";
50      } else {
51
52          int max = Integer.MIN_VALUE;
53          int index = -1;
54          for (int i = 0; i < optionCount; i++) {
```



```
53
54      @Override
55      protected void renderResult(final int position, PollOptionItemData p
56
57          final FeedListItem feedListItem = pollOptionItemData.pollItemDat
58          final Content.PollData pollData = feedListItem.getContent().getP
59          final PollItemData pollItemData = pollOptionItemData.pollItemDat
60
61          if (pollData.totalVotes() == 0 && !feedListItem.getContent().aut
62              leftRating.setVisibility(View.GONE);
63              rightRating.setVisibility(View.GONE);
64          } else {
65              leftRating.setVisibility(View.VISIBLE);
66              leftRating.setText(pollItemData.getYesText());
67              leftRating.setTextColor(mResources.getColor(pollItemData.isY
68              leftRating.setBackgroundResource(pollItemData.isYesWin() ? R
69              rightRating.setVisibility(View.VISIBLE);
70              rightRating.setText(pollItemData.getNoText());
71              rightRating.setTextColor(mResources.getColor(pollItemData.is
72              rightRating.setBackgroundResource(pollItemData.isNoWin() ? R
73          }
74      }
75
```

Build data model in background, then display the data in view

DataModel & View: DraweeController


```
65          public SEImageLoadParam build() {
66              SEImageLoadParam param = new SEImageLoadParam();
67              if (!TextUtils.isEmpty(mUserName) && TextUtils.isEmpty(mOriginUrl)) {
68                  param.mFallbackDrawable = DefaultAvatarGenerator.generateAvatar(mUserName);
69              } else {
70
71                  final String url = SEImageUrlManager.getInstance().getRemoteUrl(mOriginUrl, mWidth);
72                  param.mDraweeController = (PipelineDraweeController) Fresco.newDraweeControllerBuilder()
73                          .setAutoPlayAnimations(true)
74                          .setUri(url)
75                          .build();
76
77                  param.mUrl = url;
78                  param.mOriginUrl = mOriginUrl;
79              }
80              return param;
81          }
```

```
59      public static void loadImage(SESimpleDraweeView simpleDraweeView, @Nullable SEImageLoadParam loadParam) {
60          if (loadParam == null) {
61              simpleDraweeView.setController(null);
62              return;
63          }
64          final Drawable drawable = loadParam.getFallbackDrawable();
65          if (drawable != null) {
66              simpleDraweeView.setImageDrawable(drawable);
67          } else {
68              final DraweeController oldController = simpleDraweeView.getController();
69              if (oldController != null && oldController.equals(loadParam.getDraweeController())) {
70                  return;
71              }
72              simpleDraweeView.setController(loadParam.getDraweeController());
73          }
74      }
75  }
```

Create scrap views in background for RecyclerView


```
19  public abstract class ViewHolderBase<ItemDataType> {
20
21      protected int mLastPosition;
22      protected int mPosition = -1;
23      protected View mCurrentView;
24
25      /**
26       * create a view from resource Xml file, and hold the view that may be used in displaying data.
27       */
28      public abstract View createView(LayoutInflater layoutInflater, ViewGroup parent);
29
30      /**
31       * using the held views to display data
32       */
33      public abstract void showData(int position, ItemDataType itemData);
```

`createView` and `showData`

Create scrap views in background for RecyclerView


```
65  @   public RecyclerView.RecycledViewPool createRecycledViewPool(ViewGroup parentView, RecyclerView.RecycledViewPool recycledViewPool) {
66          if (recycledViewPool == null) {
67              recycledViewPool = new RecyclerView.RecycledViewPool();
68          }
69          for (int i = 0, nsize = mLazyCreators.size(); i < nsize; i++) {
70              final int viewType = mLazyCreators.keyAt(i);
71              LazyViewHolderCreator<ItemDataType> lazyViewHolderCreator = mLazyCreators.valueAt(i);
72              final int maxRecycledViews = lazyViewHolderCreator.getMaxRecycledViews();
73              recycledViewPool.setMaxRecycledViews(viewType, maxRecycledViews);
74              for (int j = 0; j < maxRecycledViews; j++) {
75                  RecyclerView.ViewHolder viewHolder = createViewHolder(parentView, viewType);
76                  recycledViewPool.putRecycledView(viewHolder);
77              }
78          }
79          return recycledViewPool;
80      }
```

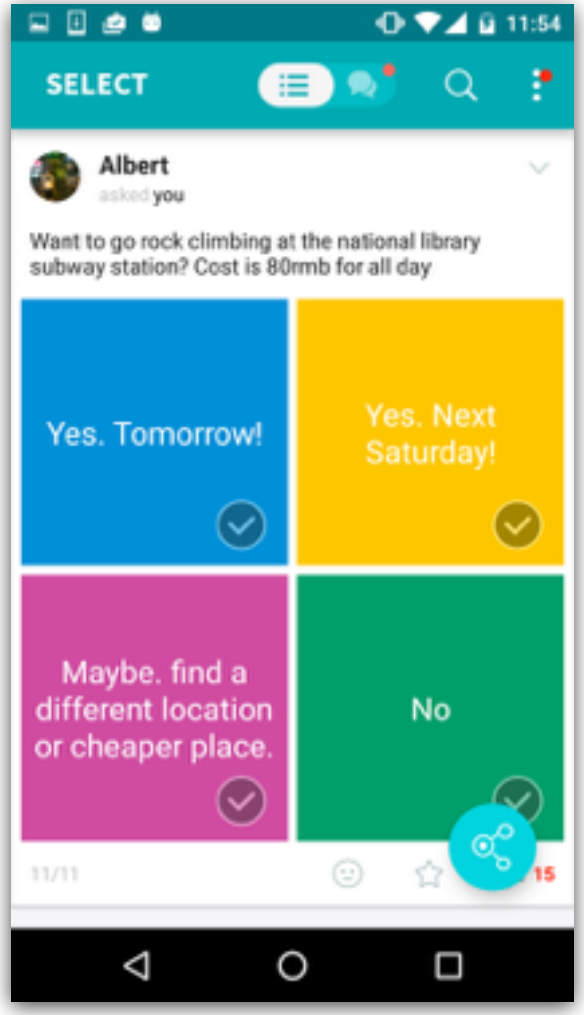
Create view in background thread ahead of time

Create scrap views in background for RecyclerView


```
57      private void setupRecyclerView() {
58          FeedListItemViewHolderManager.bindViewHolderForMultipleImage(mRecyclerViewAdapter);
59
60          LinearLayoutManager layoutManager = new GridLayoutManager(mRecyclerView.getContext(), 2);
61          mRecyclerView.setLayoutManager(layoutManager);
62          mRecyclerView.setHasFixedSize(true);
63          mRecyclerView.setAdapter(mRecyclerViewAdapter);
64          mRecyclerView.setItemAnimator(null);
65          mRecyclerView.addItemDecoration(new GridItemDecoration(ResourceProvider.get().feedListImageGutter));
66
67          final RecyclerView.RecycledViewPool pool = FeedListDataModel.getInstance().
68                  getFeedListItemViewHolderManager().
69                  getRecycledViewPoolForMultipleImage();
70          mRecyclerView.setRecycledViewPool(pool);
71
72          final int size = ResourceProvider.get().feedListImageGutterHalf;
73          mRecyclerView.setPadding(size, 0, size, 0);
74      }
```

Use the scrap views via RecycledViewPool

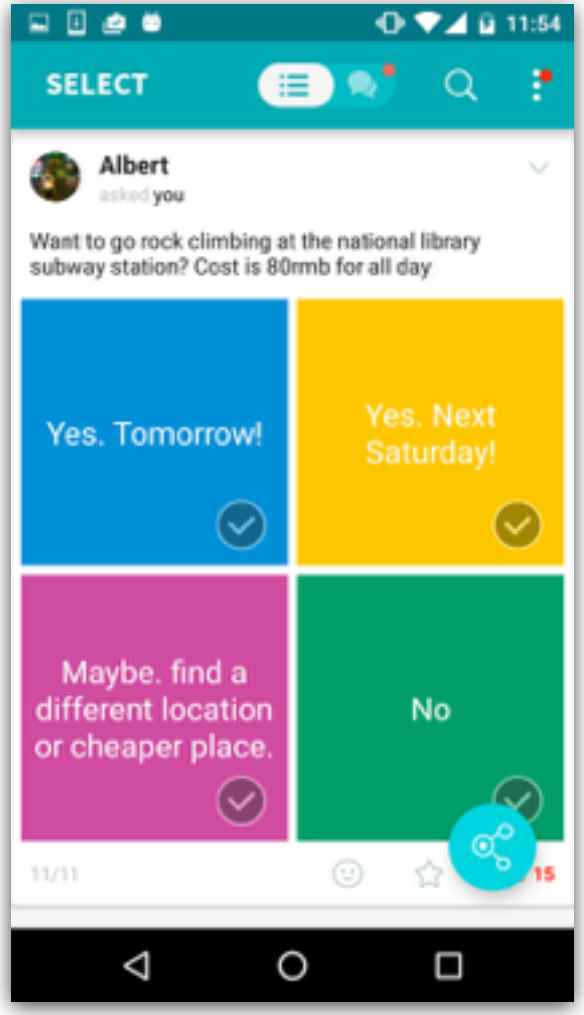
OnClickHandler issue



How to pass the data?

```
46          if (!mine) {
47              //no any select
48              if (selection == PollPost.SELECTION_NONE) {
49                  leftButton.setEnabled(true);
50                  leftButton.setOnClickListener(new View.OnClickListener() {
51                      @Override
52                      public void onClick(View view) {
53                          selectOption(poll, PollPost.SELECTION_YES);
54                      }
55                  });
56                  rightButton.setEnabled(true);
57                  rightButton.setOnClickListener(new View.OnClickListener() {
58                      @Override
59                      public void onClick(View view) {
60                          selectOption(poll, PollPost.SELECTION_NO);
61                      }
62                  });
63              //have select
64              } else {
65                  leftButton.setChecked(selection == PollPost.SELECTION_YES);
66                  rightButton.setChecked(selection == PollPost.SELECTION_NO);
67              }
68          }
```

OnClickHandler issue: Data / ViewTag

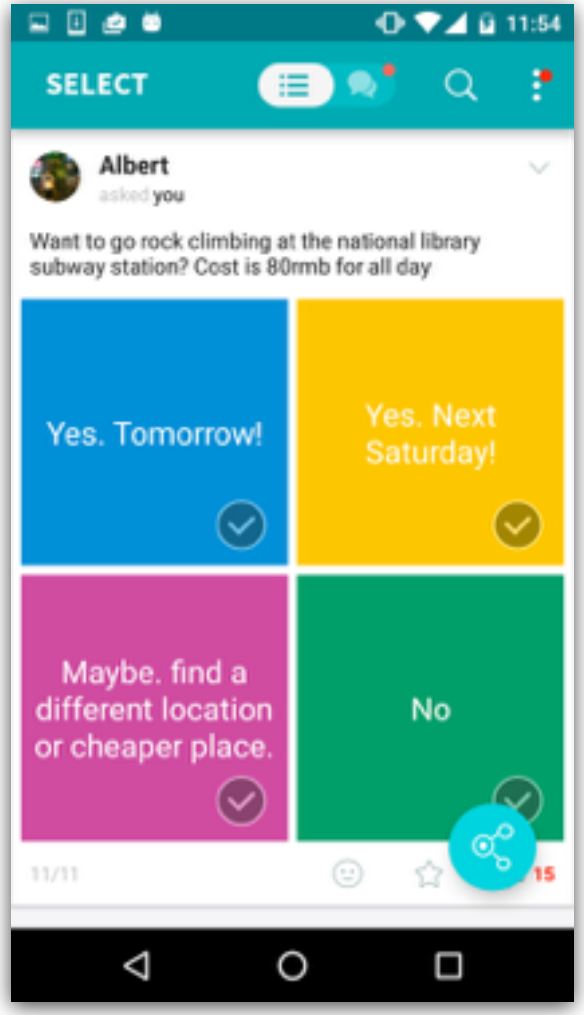


ViewTag / Data

```
public void bindToView(View view, int position) {
    ViewTagUtil.setViewTagForClick(VIEW_TAG_FOR_DATA, view, this);
}

@Nullable
public static FeedListItem fromViewTag(View view) {
    FeedListItem item = ViewTagUtil.getTag(VIEW_TAG_FOR_DATA, view, FeedListItem.class);
    return item;
}
```

OnClickHandler issue: Data / ViewTag



ViewTag / Data

```
@Override
public View createView(LayoutInflater layoutInflater, ViewGroup parent) {

    mResources = layoutInflater.getContext().getResources();
    View view = layoutInflater.inflate(R.layout.views_feed_list_item_poll_option_multiple,
false);

    // ...
    rightButton = (Button) view.findViewById(R.id.right_button);
    bindButtonCLick(rightButton);
    return view;
}

@Override
protected void renderForViewer(final int position, PollOptionItemData pollOptionItemData)
    final FeedListItem feedListItem = pollOptionItemData.pollItemData.feedListItem;
    bindOptionViewForOperate(rightButton, feedListItem, position);
}
```

Defer calculation

Defer

Defer image loading

Fresco / Glide

Defer writing log

message queue + writing thread

Principles and examples

Reduce calculation

- Consider time complexity
- Time and space trade-offs
- Cache: Size controller
- Don't do unnecessary work

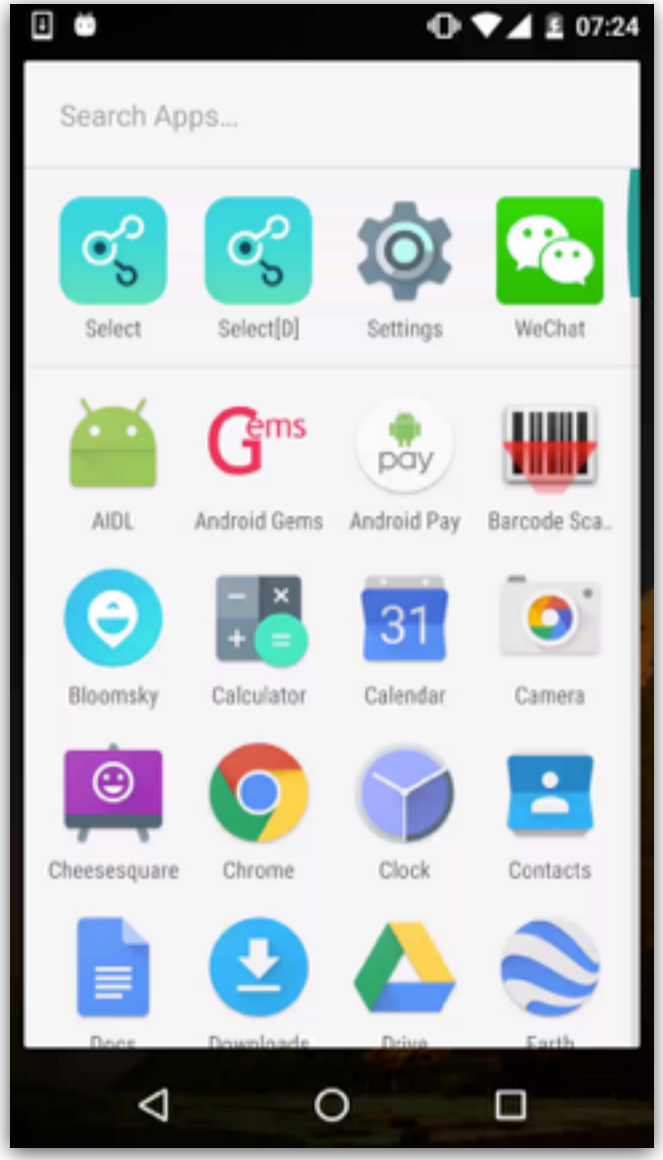
AOT calculation

- DataModel + View
- RecycledViewPool
- OnClickHandler

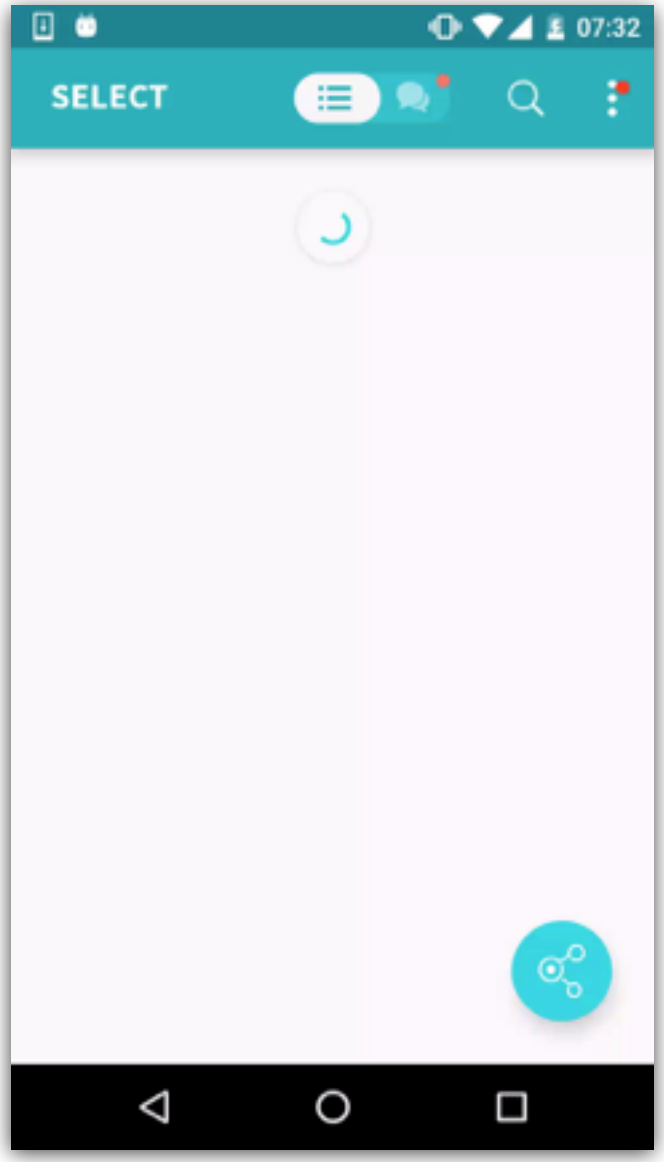
Defer calculation

- Loading image
- Writing log

https://youtu.be/qjU5kuVamu0



https://youtu.be/-n8DhbzYRaI



Challenges and solutions

Challenges

Deadline

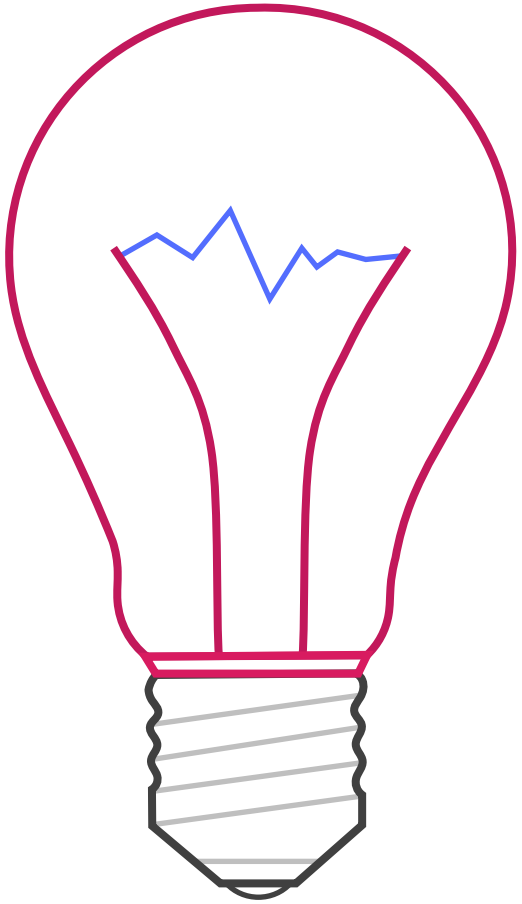
01

Team culture.
they don't care about the quality of the code

Huge Workload

02

Sometime you maybe need to rewrite the whole module.



Full of challenges

03

Full of challenges and creatives, sometimes there is totally no clue for how to make it work.

Need to know everything

04

You can not missed every detail, you should write every single line of code very carefully.

Solutions

Algorithm, Data Structure

The basic CS knowledge is very important.

Keep trying

Keep learning, keep searching, keep working hard

Change another way

or another company

Ask for help

Ask google, ask community, do ask me

Join the community!

http://join-deepint.liaohuqiu.net/

Thank You

Questions?

https://liaohuqiu.net/about/about-me/

liaohuqiu

@liaohuqiu

liaohuqiu