Improving the scheduler by making dag runs more coherent

A lot of discussion has been around scheduler issues. Some of the symptoms are requiring the start_date to align with the (cron)interval, not being able to update a DAG with a new interval, backfills interfering with normal dag runs, unable to version dags. Work arounds also have been provided eg., ignore_first_depends_on_past. Many issues on github have been created with questions around why does the scheduler do this and not this. We can hardly say it is working intuitively at the moment.

The following is a proposal for an update to the scheduler while remaining 99% backwards compatible. Code (PR-1431) is fully working and all tests pass on travis. Now I am looking for real world testers, feedback - Paul I understood you are also working on something on the scheduler? - etc. :-).

SUMMARY

Model changes
- TaskInstances now always have an associated DagRun
- TaskInstances can still be run without a real DagRun but they need the special "-1" DagRun id.
- Backfills also create DagRuns

Scheduler
- Scheduler is aware of backfills
- Scheduler will fast forward beyond backfill if needed while maintaining the correct interval
- Scheduler will adjust start date for interval (no more aligning needed)

Backfills
- Backfills can update the past, but will maintain lineage
- Backfills insert themselves in between scheduled dag runs if needed, making depend_on_past work with arbitrary inserted backfills

Where might there be issues:
- If you are creating TaskInstances inside DAGs yourself or in Operators you might need to update those to take into account a dag_run_id. For Operators have a look at the PythonOperator how to update it.
- If you are issuing "airflow run xxx" *yourself*, thus outside airflow: for now Airflow will set your dag_run_id to -1, but is this really what you want? Please let me know the use cases and discuss.
- For depends_on_past I check on the DagRun level, it might be required to move this to the TaskInstance level. Not a big change and we can even support both!

BACKGROUND
If you consider DAGs a description of coherence work and Tasks (as defined in Airflow) a description of work, then DagRuns are the instantiation of DAGS in a point of time. TaskInstances are then the instantiations of Tasks in a point of time. To maintain the coherence between TaskInstances a DagRun needs to be aware of the TaskInstances it has. So far so good. Airflow does this quite well and as long as you don't try to do something fancy like updating you interval with depend_on_past your tasks will run happily. Now enter backfills. Backfills allow you to either create or alter history. It does this by arbitrarily inserting tasks into the time line and completely disregarding the scheduler. Also the scheduler does not know about backfills leading to other issues.

If we would maintain a time line of DagRuns this would solve the above mentioned issues, improve lineage and pave the way for DAG versioning. It would also simplify the code in the future by moving a lot of logic to DagRun. While one might argue that currently "dag_id + execution_date" already does this for both DagRun and TaskInstances this forgoes the issue that backfills create in the scheduler, in lineage and in versioning in addition this would not allow you to solve the moving interval easily. With backfills you would be able to run an updated dag that changes the past, but how to answer what version of the DAG you ran when? Depending on past in a backfill with a new task is also quite hard.

IMPLEMENTATION
DagRuns now maintain a "previous" property. Previous points to the previous dag run by id, if previous is None it is considered the first DagRun. The scheduler will set the previous property if it detects a previous run. A previous run can be a scheduled run or a backfill run. If will adhere to the schedule_interval, but it will fast forward beyond the latest execution date of either the last scheduled run or last backfill run whatever comes later. So at the moment the scheduler will not fill in the blanks for you if there is a gap between the scheduled run and a backfill run of more than one interval.

Backfill will create a DagRun and insert it into the timeline, ie. it will update the previous property of a scheduled DagRun if it is set in the future seen from the Backfill DagRun. If it encounters a DagRun at the same execution date for the same dag_id it will re-own the tasksinstances and set the state of the other DagRun to "overridden", essentially orphaning the other DagRun but keeping its record around for auditing purposes.

TaskInstances now maintain a reference to a DagRun, which cannot be None. If it is created with None it will be set to the special "-1" DagRun ID. DagRun IDs that are "None" will lead to an integrity error at the database level. "-1" DagRun ID TaskInstances are treated as they are now. However Airflow itself will not create such TaskInstances anymore so they come from outside. In the context of the DARGS being the description of "coherence of work" I have a lot of difficulty understanding having work available that is not coherent :-).