# zupakka (ddm-akka homework)

Daniel Engelbach, Jonathan Wendt, Desirée Wenk

# startup: Dependency Miner

1. files get read and saved into ArrayLists
2. once finished the line-wise structure gets converted into a column-wise structure
3. a Task-List gets created, consisting of UniqueTasks
    1. each column should ideally only contain each value once to save on computation and bandwidth further down the line

# handling of workers

- once the DependencyMiner receives a RegistrationMessage from a Worker, a reference to it gets saved into a list of idle workers
- if a task from the beforementioned TaskList get assigned, the corresponding Worker reference is moved to the busyWorkers-list
  - analog every time a task gets finished (ResultMessage to the DependencyMinder)
- assigned tasks are removed from the TaskList
  - a seperate reference of them is kept in order to be able to reassign it in case a running worker dies

# message handling

- basically for each type of message (UniqueTasks, INDTasks, correspondig results, …) a different Message with the needed data is implemented
- given the possible size of those messages the LargeMessageProxies are used to transfer messages between the DependencyMinder and workers

# IND discovery

# unique tasks

- The UniqueTasks (mainly consisting of indices as well as the raw data) are assigned to ready workers
  - if a certain percentage of the first 10k entries are redundant the filtering continues, finishing with returning only unique values to the DependencyMiner
  - otherwise the whole dataset gets send back

# IND tasks

1. once a UniqueTask result is received that column is marked as ready for further processing and INDTasks containing that column are generated
2. INDTasks contain indices as well as ArrayLists of the raw content of two columns A and B
3. the worker (INDFinder-subworker) converts the Lists into Maps in order to speed up contain-checks
4. it is looped through both columns checking for containment of each value in the other
   - if a value isn‘t included in the other column the loop breaks and the corresponding INDResult gets marked as false
   - once finished the results get sent back to the DependencyMiner

# result collection and finishing

- everytime a positive INDResult is received it gets written to the results-file using the already implemented ResultCollector
- once the TaskList is empty and there are no busyWorkers left the program shuts down

# comments - ATTENTION

- besides working properly there‘s still a lot of room for improvement performance-wise as in its current state the main bottleneck is network-bandwidth
- furthermore the Master instance is quite ressource heavy, not being able to run on less than ~3,5GB of RAM, the workers are fine with less than 2GB though