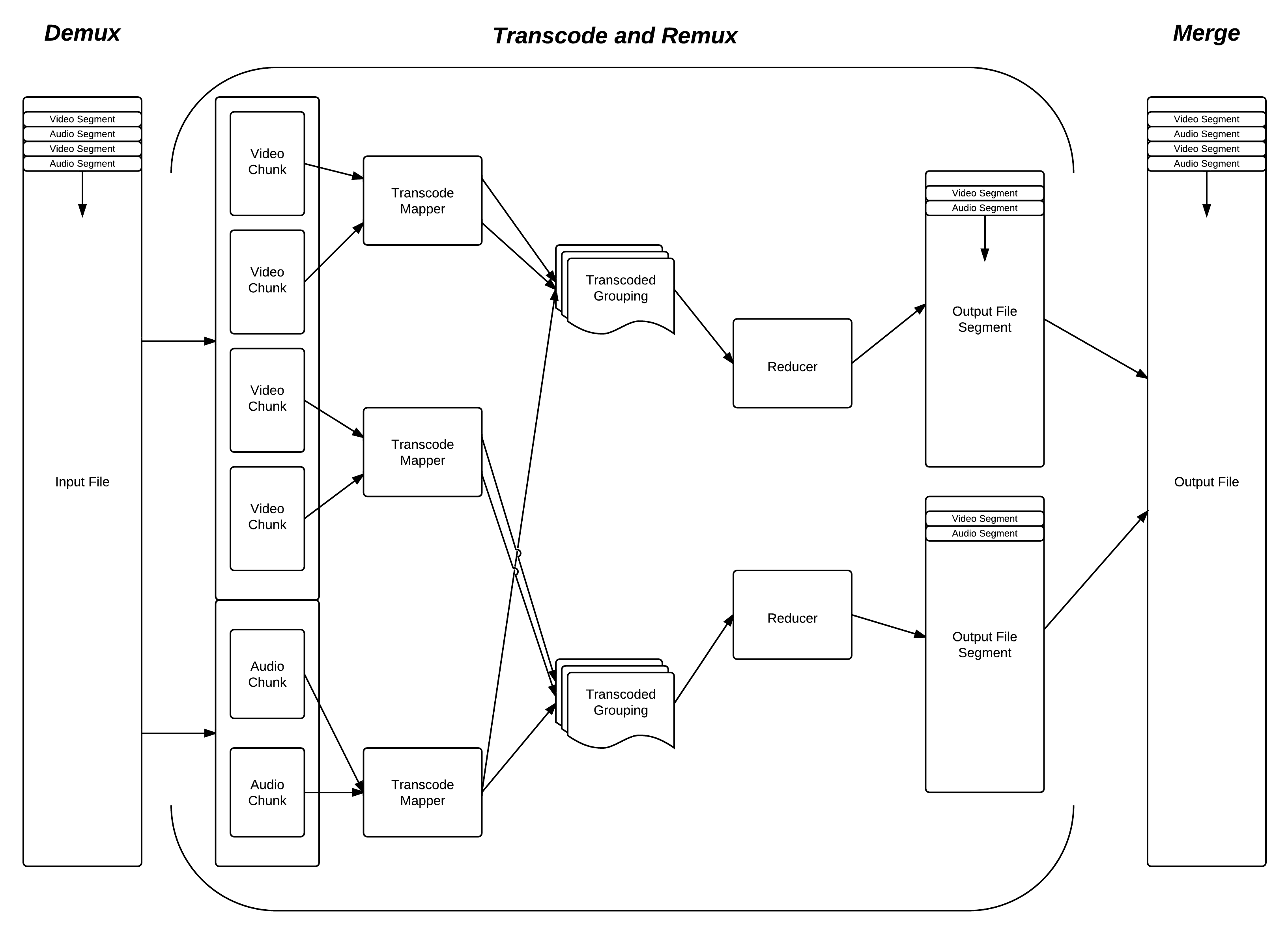
FFmpeg-MR Project Plan

This project plan will initially go over the system design to give an overview of what the project aims to produce. It will then detail the technical implementation phases of work that will be undertaken, identify any risks, and detail the documents that are to be produced along with the system and evaluation that will take place over the course of the project.

# System Overview



*(It is assumed that the Project Specification has been read. For more detail on the system, also see the Initial Design Document)*

The system is made up of several high level phases; ‘demux’, ‘transcode’, ‘remux’, and merge. In the ‘demux’ phase we split up the input file into its constituent streams, convert it to a Hadoop friendly storage format, and load it into the distributed file system. We then process splits of this data across designated ‘mapper’ machines, in the ‘transcode’ phase. We then group the data together by its timestamp in the input, in a process the Hadoop system calls the ‘shuffle’, and pass the new grouped data into ‘reducers’ where the ‘remux’ phase occurs. The ‘remux’ puts the constituent data streams back into its ordered format, with timestamps and encoding data. The chunks then produced in the ‘remux’ by the various reducer machines are then merged together ‘on the fly’ when the data it requested back out of the distributed file system.

# Technical Implementation Phases

*Note: Week numbers start Oct 17th.*

Splitting the implementation up into phases allows for the technical implementation to stop at any of the phase boundaries with a workable solution. As one of the most important parts of the project overall is the project report – ensuring that the technical side of the project can stop in time for a report to written is critical. If the technical work is to slow down, or come across unforeseen circumstances, then this will allow the final report to be produced with a solution that works.

**Phase 1 - Week 3**

* Implement the basic FFmpeg <-> Java interface for transcoding
  + Ability to transcode a MJPEG stream into H.264 using MR
  + Ability to transcode a MP3 stream into AAC using MR
* Stream will be manually generated for input
* Single mapper (so no splitting), single reduce (so no partitioning).
* Gets the FFmpeg interoperability basics working from Java, and acts as a proof of concept for the main work.

**Phase 2 - Week 5**

* Ability to run several mappers (so we can actually be quicker and run in paralell!)
* Code that generates the streams as Hadoop SequenceFiles from an input file using ffmpeg to be written.
* Single reduce (so no partitioning).
* Many machine - test with several to ensure that map splits are working.
* Input will be a file to be demuxed, but output will still be raw streams.
* Write up the design document.

**Phase 3 - Week 8**

* Add support for getting initial input from Amazon S3, and writing output to Amazon S3.
* Add support for submitting the job to a Elastic MapReduce cluster using Apache Whirr.
* Ability to mux different streams together in the reducers.
* Ability to use several reducers by implementing a partitioning function that spreads the work out fairly.
* Output is still in segments, corresponding to the number of reducers we chose. 1 reducer won't be the best performing, but it will output a complete file.

**Phase 4 - Week 10**

* Job submission API (either running on a controller, or maybe just on the machine that is submitting data).
  + Input file
  + Desired outputs
  + API figures out what to actually execute and gets on with it
* Support for HTTP Live Streaming as an output from the reducers

**Phase 5 - (Week 12?)**

* Web interface for job submission and monitoring.
  + Displays detailed information about what operations are being performed and where so we can explain the system better.
  + Shows the execution plan that the API comes up with when you ask for output so that the user knows exactly what is going to happen.
* Support for 'chunking' the input into smaller jobs, so that the output can be used earlier by external systems.
  + Also allows for input to be 'streamed' and then sent into the system in a decent size chunk (there will be a delay on the stream).
  + Lots of logic and code can be taken from the HTTP Live Streaming support in phase 4 - as it is quite similar from an output POV.
  + This would also allow for jobs to be submitted by remote clients using the java API, and the input be sent directly to the cluster in chunks, such that encoding happens sooner.

**Phase 6 - (Week 14?)**

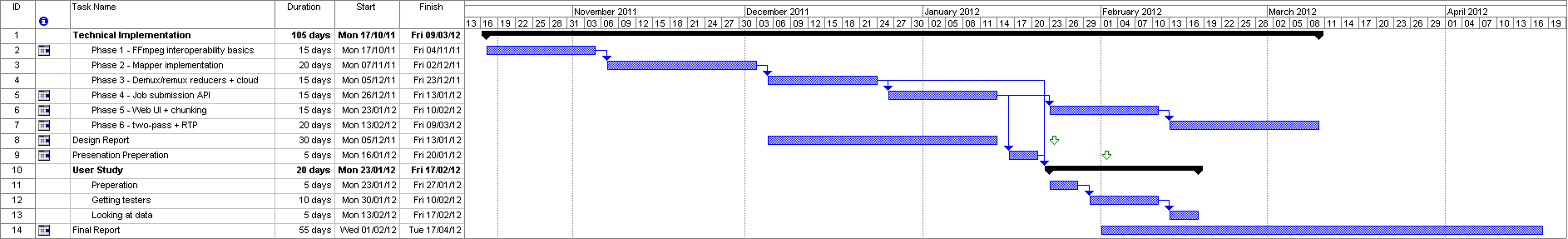
* Two-pass video transcoding
  + The initial step of splitting the streams, and generating a sequence file to be turned into an MR job.
  + This MR job will also use ffmpeg to perform a first pass on the streams that support it and generate complexity data.
  + This data is then used in the second map phase (the original transcode) to help the encoder to decide where is best to use a higher bitrate, and where not.
* Support for RTP input streams to be added on the back of the chunked jobs.
  + This would require a remote machine to act as a controller (this could be the Hadoop job-tracker however).

# Known Risks

The main risk area from a technical point of view is the use of FFmpeg and bridge of native code across to Java using JNI. FFmpeg, whilst being a mature project (since 2004) and an active one (last release 12 days ago!), it hasn’t got extensive developer documentation, and its usage will involve an amount of experimentation to achieve the results I’m looking for. I’m sure of its ability to transcode and ‘mux’ in my system – they release an application (FFmpeg itself) that has this functionality and relies on the same libraries I will be using as a base.

Using JNI to write this interface is also a point of risk – it is an unfamiliar framework – renowned for its complicated implementation and ease at which the stability of the JVM can be compromised. To be sure I could work productively in this environment – the first phase of the project is aimed at as proof of concept at getting JNI, and FFmpeg to work, some of which has already been proven to work correctly in my early work.

# Project Plan



Work on the technical implementation has already begun. Once the first two stages of this have been completed (early/mid-December) – and we have a semi-working system – the work on the formal design report will begin. This will continue as the next two phases of the implementation take place, so that upon my return for the start of term two, the design report and the first 4 phases of the implementation will be complete.

Some time will then be used to prepare for the presentation, using content from the design report and the real working system. Once the presentation has been completed, planning for the user study will begin. Volunteers will be called for and selected, and the study will take place. Whilst this quite lengthy process is taking place, some more work on the later phases of the project (‘Advanced Objectives’) will take place, but with less importance that the user study. Preliminary work on the report will start to take place, in mid-February, bringing together content from the literature survey, writing up the analysis of the user study on quality vs. speed, and on the specifics of the design and implementation that took place in prior months.

Lots of time (from mid-March onwards) is then devoted to the report, and finishing off the final phases, along with spare time for both other commitment (such as exams) and slack time in case of any issues.