A Word Frequency

This section contains a program that counts the number of occurrences of each unique word in a set of input files specified on the command line.

```
#include "mapreduce/mapreduce.h"

// User's map function
class WordCounter : public Mapper {
 public:
  virtual void Map(const MapInput& input) {
    const string& text = input.value();
    const int n = text.size();
    for (int i = 0; i < n; ) {
      // Skip past leading whitespace
      while ((i < n) && isspace(text[i]))
        i++;

      // Find word end
      int start = i;
      while ((i < n) && !isspace(text[i]))
        i++;
      if (start < i)
        Emit(text.substr(start,i-start),"1");
    }
  }
};
REGISTER_MAPPER(WordCounter);

// User's reduce function
class Adder : public Reducer {
  virtual void Reduce(ReduceInput* input) {
    // Iterate over all entries with the
    // same key and add the values
    int64 value = 0;
    while (!input->done()) {
      value += StringToInt(input->value());
      input->NextValue();
    }

    // Emit sum for input->key()
    Emit(IntToString(value));
  }
};
REGISTER_REDUCER(Adder);

int main(int argc, char** argv) {
  ParseCommandLineFlags(argc, argv);

  MapReduceSpecification spec;

  // Store list of input files into "spec"
  for (int i = 1; i < argc; i++) {
    MapReduceInput* input = spec.add_input();
    input->set_format("text");
    input->set_filepattern(argv[i]);
    input->set_mapper_class("WordCounter");
  }

  // Specify the output files:
  //    /gfs/test/freq-00000-of-00100
  //    /gfs/test/freq-00001-of-00100
  //    ...
  MapReduceOutput* out = spec.output();
  out->set_filebase("/gfs/test/freq");
  out->set_num_tasks(100);
  out->set_format("text");
  out->set_reducer_class("Adder");

  // Optional: do partial sums within map
  // tasks to save network bandwidth
  out->set_combiner_class("Adder");

  // Tuning parameters: use at most 2000
  // machines and 100 MB of memory per task
  spec.set_machines(2000);
  spec.set_map_megabytes(100);
  spec.set_reduce_megabytes(100);

  // Now run it
  MapReduceResult result;
  if (!MapReduce(spec, &result)) abort();

  // Done: 'result' structure contains info
  // about counters, time taken, number of
  // machines used, etc.

  return 0;
}
```