Assignment 2

Multiple workers and data parallelism

Below is the modified code. Important points:

- To have only worker 0 generate input data, put an `if` statement around the inner for-loop to check the worker index.
- To partition the data across workers during all rounds, add `exchange` between `input_from` and `inspect` to route the input tuples solely based on the **second** value.

```
extern crate timely;
use timely::dataflow::{InputHandle, ProbeHandle};
use timely::dataflow::operators::{Input, Exchange, Inspect, Probe};
fn main() {
    // 1) Instantiate a computation pipeline by chaining operators
    timely::execute_from_args(std::env::args(), |worker| {

        let index = worker.index();
        // create opaque handles to feed input and monitor progress
        let mut input = InputHandle::new();
        let mut probe = ProbeHandle::new();

        worker.dataflow(|scope| {
            scope.input_from(&mut input)
            .exchange(|(_round, num)| *num)
            .inspect(move |(round, num)| println!("round: #{}\tnum:
{}\tworker: {}", round, num, index))
            .probe_with(&mut probe);
        });

        // 2) Push data into the dataflow and allow computation to run
        for round in 0..10 {
            if index == 0 {
                for j in 0..round + 1 {
                    input.send((round, j));
                }
            }
            // advance input and instruct the workers to do work
            input.advance_to(round + 1);
            while probe.less_than(input.time()) {
                worker.step();
            }
        }
    }).unwrap();
}
```

Executing with 2 workers using `cargo run --release -- -w2`:

```
    Finished release [optimized] target(s) in 0.02s
    Running `target/release/timely-playground -w2`
round: #0       num: 0  worker: 0
round: #1       num: 0  worker: 0
round: #1       num: 1  worker: 1
round: #2       num: 0  worker: 0
round: #2       num: 2  worker: 0
round: #2       num: 1  worker: 1
round: #3       num: 1  worker: 1
round: #3       num: 3  worker: 1
round: #3       num: 0  worker: 0
round: #3       num: 2  worker: 0
round: #4       num: 0  worker: 0
round: #4       num: 2  worker: 0
round: #4       num: 4  worker: 0
round: #4       num: 1  worker: 1
round: #4       num: 3  worker: 1
round: #5       num: 0  worker: 0
round: #5       num: 2  worker: 0
round: #5       num: 4  worker: 0
round: #5       num: 1  worker: 1
round: #5       num: 3  worker: 1
round: #5       num: 5  worker: 1
round: #6       num: 0  worker: 0
round: #6       num: 2  worker: 0
round: #6       num: 4  worker: 0
round: #6       num: 6  worker: 0
round: #6       num: 1  worker: 1
round: #6       num: 3  worker: 1
round: #6       num: 5  worker: 1
round: #7       num: 0  worker: 0
round: #7       num: 2  worker: 0
round: #7       num: 4  worker: 0
round: #7       num: 6  worker: 0
round: #7       num: 1  worker: 1
round: #7       num: 3  worker: 1
round: #7       num: 5  worker: 1
round: #7       num: 7  worker: 1
round: #8       num: 0  worker: 0
round: #8       num: 2  worker: 0
round: #8       num: 4  worker: 0
round: #8       num: 6  worker: 0
round: #8       num: 8  worker: 0
round: #8       num: 1  worker: 1
round: #8       num: 3  worker: 1
round: #8       num: 5  worker: 1
round: #8       num: 7  worker: 1
round: #9       num: 0  worker: 0
round: #9       num: 2  worker: 0
round: #9       num: 4  worker: 0
round: #9       num: 6  worker: 0
round: #9       num: 8  worker: 0
round: #9       num: 1  worker: 1
```

```
round: #9       num: 3  worker: 1
round: #9       num: 5  worker: 1
round: #9       num: 7  worker: 1
round: #9       num: 9  worker: 1
```

Word Count

Run the following code with `cargo run --release -- -w4` with 4 workers:

```
extern crate timely;

// import all necessary modules
use std::io::{BufReader, BufRead};
use std::fs::File;
use std::hash::{Hash, Hasher};
use std::collections::hash_map::DefaultHasher;

use timely::dataflow::{InputHandle, ProbeHandle};
use timely::dataflow::operators::{Input, Map, Inspect, Probe};
use timely::dataflow::operators::aggregation::Aggregate;

fn hash_str<T: Hash>(t: &T) -> u64 {
    let mut s = DefaultHasher::new();
    t.hash(&mut s);
    s.finish()
}

fn main() {
    // 1) Instantiate a computation pipeline by chaining operators
    timely::execute_from_args(std::env::args(), |worker| {

        let index = worker.index();
        // create opaque handles to feed input and monitor progress
        let mut input = InputHandle::new();
        let mut probe = ProbeHandle::new();

        worker.dataflow(|scope| {
           scope.input_from(&mut input)
                .flat_map(|text: String|
                   text.split_whitespace()
                       .map(move |word| (word.to_owned(), 1))
                       .collect::<Vec<_>>()
                )
                .aggregate(
                   // fold: combines new data with existing state
                   |_key, val, agg| { *agg += val; },
                   // emit: produce output from state
                   |key, agg: i64| (key, agg),
                   // hash: route data according to a key
                   |key| hash_str(key)
                )
                .inspect(move |(word, count)| println!("worker: #{}\tword: 
{}\tcount: {}", index, word, count))
                .probe_with(&mut probe);
        });

        let path = format!("/home/zhifei/repo/dspa/session-2-timely/input-
{}.txt", index);
        let file = File::open(path).expect("Input data not found in CWD");
        let buffered = BufReader::new(file);
        // send input line-by-line
        let mut total_lines = 0;
        for line in buffered.lines() {
            input.send(line.unwrap());
            total_lines = total_lines + 1;
        }
        // advance input and process
        input.advance_to(total_lines + 1);
        while probe.less_than(input.time()) {
            worker.step();
        }
    }).unwrap();
}
```

See `input-{0..4}.txt` and `heyjude.out` in the appendix for its input and output.

This wordcount is different from the `SocketWindowWordCount` example in Assignment 1 in that

1. It reads text from textfiles, while `SocketWindowWordCount` reads text from a socket,
2. For every word it counts the total occurrences in all files, while `SocketWindowWordCount` counts the occurrences of different words inside each time window of 5 seconds.

To change the program to read text, send inputs, and perform computation per 5 lines of text each time instead of all at once, modify the last part so that it

1. reads 5 lines (or all remaining lines if less than 5) into a string,
2. sends the string to the input as a whole,
3. advances input and processes these lines.

The code is the following. The output is `heyjude_5lines.out` in the appendix. In each round, every worker sends 5 lines (or less when meeting file end) to the input, and then output the counts of the words (in these lines) that it is responsible for. The computation is triggered once 5 lines has been sent (instead of having sent all lines), so it enables continuous computation.

Since the files are not updating, a infinite loop does not much sense, but if it is to be replaced by a `TcpStream`, the loop to read lines from the files should also be replaced by a infinite loop that continuously read lines from the stream. If each worker reads from a different stream and a stream is significantly slower than others, this may cause a performance issue because that worker needs to wait for input instead of participating in the aggregating process triggered by others.

```
extern crate timely;

use std::io::{BufReader, BufRead};
use std::fs::File;
use std::hash::{Hash, Hasher};
use std::collections::hash_map::DefaultHasher;

use timely::dataflow::{InputHandle, ProbeHandle};
use timely::dataflow::operators::{Input, Map, Inspect, Probe};
use timely::dataflow::operators::aggregation::Aggregate;

fn hash_str<T: Hash>(t: &T) -> u64 {
    let mut s = DefaultHasher::new();
    t.hash(&mut s);
    s.finish()
}

fn main() {
    // 1) Instantiate a computation pipeline by chaining operators
    timely::execute_from_args(std::env::args(), |worker| {

        let index = worker.index();
        // create opaque handles to feed input and monitor progress
        let mut input = InputHandle::new();
        let mut probe = ProbeHandle::new();

        worker.dataflow(|scope| {
           scope.input_from(&mut input)
                .flat_map(|text: String|
                   text.split_whitespace()
                       .map(move |word| (word.to_owned(), 1))
                       .collect::<Vec<_>>()
                )
                .aggregate(
                   // fold: combines new data with existing state
                   |_key, val, agg| { *agg += val; },
                   // emit: produce output from state
                   |key, agg: i64| (key, agg),
                   // hash: route data according to a key
                   |key| hash_str(key)
                )
                .inspect(move |(word, count)| println!("worker: #{}\tword: 
{}\tcount: {}", index, word, count))
                .probe_with(&mut probe);
        });

        let path = format!("/home/zhifei/repo/dspa/session-2-timely/input-
{}.txt", index);
        let file = File::open(path).expect("Input data not found in CWD");
        let mut buffered = BufReader::new(file);

        let mut total_lines = 0;
        let mut lines = String::new();
        let mut round = 0;
        while buffered.read_line(&mut lines).expect("Unable to read a
line") > 0 {
            total_lines += 1;
            if total_lines % 5 == 0 {
                println!("Worker #{} Input Lines {}~{}:", index,
total_lines - 4, total_lines);
                // send buffered 5 lines to input, create a new empty
buffer
                input.send(lines);
                lines = String::new();
                // advance input and process
                input.advance_to(round + 1);
                while probe.less_than(input.time()) {
                    worker.step();
                }

                // next round
                round += 1;
            }
        }
        // the last few lines
        if total_lines % 5 != 0 {
            println!("Worker #{} Final Input Lines {}~{}:", index,
total_lines - total_lines % 5 + 1, total_lines);
            input.send(lines);

            // advance input and process
            input.advance_to(round + 1);
            while probe.less_than(input.time()) {
                worker.step();
            }
        }
    }).unwrap();
}
```

Appendix

`input-0.txt`:

```
Hey, Jude, don't make it bad
Take a sad song and make it better
Remember to let her into your heart
Then you can start to make it better

Hey, Jude, don't be afraid
You were made to go out and get her
The minute you let her under your skin
Then you begin to make it better

And anytime you feel the pain,
Hey, Jude, refrain
Don't carry the world upon your shoulders
For well you know that it's a fool
Who plays it cool
By making his world a little colder

Nah, nah nah, nah nah, nah nah, nah nah
```

`input-1.txt`:

```

Hey, Jude, don't let me down
You have found her, now go and get her
Remember to let her into your heart
Then you can start to make it better

So let it out and let it in,
Hey, Jude, begin
You're waiting for someone to perform with
And don't you know that it's just you,
Hey, Jude, you'll do
The movement you need is on your shoulder

Nah, nah nah, nah nah, nah nah, nah nah yeah
```

`input-2.txt`:

```

Hey, Jude, don't make it bad
Take a sad song and make it better
Remember to let her under your skin
Then you'll begin to make it better, better, better, better, better... oh!

Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (Jude)
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (yeah, yeah, yeah)
```

`input-3.txt`:

```

Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (don't make it bad, Jude)
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (take a sad song and make it better)
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (oh, Jude)
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (Jude, hey, Jude, whoa)
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude (ooh)
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude
Nah, nah nah, nah nah, nah, nah, nah nah,
Hey, Jude [fade out]
```

heyjude.out:

```
worker: #1      word: You're    count: 1
worker: #1      word: skin      count: 2
worker: #1      word: into      count: 2
worker: #1      word: You       count: 2
worker: #1      word: Nah,      count: 18
worker: #1      word: refrain   count: 1
worker: #1      word: his       count: 1
worker: #1      word: your      count: 6
worker: #1      word: Hey,      count: 23
worker: #1      word: upon      count: 1
worker: #1      word: And       count: 2
worker: #1      word: shoulders count: 1
worker: #1      word: hey,      count: 1
worker: #1      word: go        count: 2
worker: #1      word: begin     count: 3
worker: #1      word: under     count: 2
worker: #1      word: song      count: 3
worker: #1      word: heart     count: 2
worker: #1      word: fool      count: 1
worker: #1      word: little    count: 1
worker: #1      word: perform   count: 1
worker: #1      word: movement  count: 1
worker: #1      word: Remember  count: 3
worker: #1      word: have      count: 1
worker: #0      word: know      count: 2
worker: #0      word: better)   count: 1
worker: #0      word: For       count: 1
worker: #0      word: oh!       count: 1
worker: #0      word: her       count: 6
worker: #0      word: colder    count: 1
```

```
worker: #0      word: found     count: 1
worker: #0      word: afraid    count: 1
worker: #0      word: me        count: 1
worker: #0      word: yeah,     count: 1
worker: #0      word: nah,      count: 86
worker: #0      word: were      count: 1
worker: #0      word: making    count: 1
worker: #0      word: cool      count: 1
worker: #0      word: someone   count: 1
worker: #0      word: and       count: 6
worker: #0      word: better    count: 5
worker: #0      word: made      count: 1
worker: #0      word: Jude)     count: 2
worker: #0      word: you,      count: 1
worker: #0      word: Take      count: 2
worker: #0      word: world     count: 2
worker: #0      word: pain,     count: 1
worker: #0      word: well      count: 1
worker: #0      word: By        count: 1
worker: #0      word: is        count: 1
worker: #0      word: on        count: 1
worker: #0      word: Jude,     count: 8
worker: #0      word: (oh,      count: 1
worker: #0      word: waiting   count: 1
worker: #0      word: sad       count: 3
worker: #0      word: bad,      count: 1
worker: #0      word: to        count: 9
worker: #0      word: yeah)     count: 1
worker: #0      word: it's      count: 2
worker: #0      word: anytime   count: 1
worker: #0      word: (ooh)     count: 1
worker: #0      word: down      count: 1
worker: #3      word: plays     count: 1
worker: #3      word: out       count: 2
worker: #3      word: So        count: 1
worker: #3      word: the       count: 2
worker: #3      word: you       count: 8
worker: #3      word: get       count: 2
worker: #3      word: better... count: 1
worker: #3      word: Don't     count: 1
worker: #2      word: start     count: 2
worker: #2      word: whoa)     count: 1
worker: #2      word: (Jude)    count: 1
worker: #2      word: in,       count: 1
worker: #2      word: Who       count: 1
worker: #2      word: nah       count: 58
worker: #2      word: (Jude,    count: 1
worker: #2      word: you'll    count: 2
worker: #2      word: do        count: 1
worker: #2      word: yeah      count: 1
worker: #2      word: carry     count: 1
worker: #2      word: better,   count: 4
worker: #2      word: be        count: 1
worker: #2      word: it        count: 13
```

```
worker: #2      word: out]      count: 1
worker: #2      word: with      count: 1
worker: #2      word: (don't    count: 1
worker: #2      word: now       count: 1
worker: #2      word: The       count: 2
worker: #2      word: Jude      count: 16
worker: #2      word: Then      count: 4
worker: #2      word: that      count: 2
worker: #2      word: (yeah,    count: 1
worker: #2      word: need      count: 1
worker: #3      word: [fade     count: 1
worker: #3      word: don't     count: 5
worker: #3      word: make      count: 10
worker: #3      word: (take     count: 1
worker: #3      word: her,      count: 1
worker: #3      word: for       count: 1
worker: #3      word: feel      count: 1
worker: #3      word: let       count: 7
worker: #3      word: bad       count: 2
worker: #3      word: shoulder  count: 1
worker: #3      word: minute    count: 1
worker: #3      word: can       count: 2
worker: #3      word: just      count: 1
worker: #3      word: a count: 5
```

heyjude_5lines.out:

```
Worker #3 Input Lines 1~5:
Worker #1 Input Lines 1~5:
Worker #2 Input Lines 1~5:
Worker #0 Input Lines 1~5:
worker: #3      word: let       count: 4
worker: #1      word: go        count: 1
worker: #1      word: Hey,      count: 5
worker: #1      word: heart     count: 2
worker: #1      word: Nah,      count: 3
worker: #1      word: your      count: 3
worker: #1      word: You       count: 1
worker: #1      word: skin      count: 1
worker: #1      word: under     count: 1
worker: #1      word: have      count: 1
worker: #1      word: into      count: 2
worker: #1      word: begin     count: 1
worker: #1      word: Remember  count: 3
worker: #1      word: song      count: 2
worker: #3      word: get       count: 1
worker: #3      word: bad       count: 2
worker: #3      word: make      count: 8
worker: #3      word: a count: 2
worker: #3      word: don't     count: 3
worker: #3      word: can       count: 2
worker: #3      word: you       count: 2
```

```
worker: #3      word: her,      count: 1
worker: #3      word: better... count: 1
worker: #2      word: nah       count: 9
worker: #2      word: Then      count: 3
worker: #2      word: Jude      count: 2
worker: #2      word: better,   count: 4
worker: #2      word: now       count: 1
worker: #2      word: start     count: 2
worker: #2      word: it        count: 8
worker: #2      word: (don't    count: 1
worker: #2      word: you'll    count: 1
worker: #0      word: nah,      count: 15
worker: #0      word: sad       count: 2
worker: #0      word: better    count: 4
worker: #0      word: bad,      count: 1
worker: #0      word: Jude)     count: 1
worker: #0      word: Take      count: 2
worker: #0      word: oh!       count: 1
worker: #0      word: me        count: 1
worker: #0      word: and       count: 3
worker: #0      word: her       count: 4
worker: #0      word: down      count: 1
worker: #0      word: found     count: 1
worker: #0      word: Jude,     count: 3
worker: #0      word: to        count: 6
Worker #2 Input Lines 6~10:
Worker #1 Input Lines 6~10:
Worker #3 Input Lines 6~10:
Worker #0 Input Lines 6~10:
worker: #2      word: that      count: 1
worker: #2      word: nah       count: 15
worker: #2      word: you'll    count: 1
worker: #2      word: do        count: 1
worker: #2      word: The       count: 1
worker: #2      word: it        count: 4
worker: #2      word: (Jude,    count: 1
worker: #2      word: be        count: 1
worker: #2      word: in,       count: 1
worker: #2      word: with      count: 1
worker: #2      word: whoa)     count: 1
worker: #2      word: Then      count: 1
worker: #2      word: Jude      count: 5
worker: #3      word: for       count: 1
worker: #3      word: (take     count: 1
worker: #3      word: make      count: 2
worker: #3      word: a count: 1
worker: #3      word: minute    count: 1
worker: #3      word: you       count: 3
worker: #3      word: So        count: 1
worker: #3      word: don't     count: 2
worker: #3      word: just      count: 1
worker: #3      word: let       count: 3
worker: #3      word: get       count: 1
worker: #3      word: out       count: 2
worker: #0      word: sad       count: 1
worker: #0      word: were      count: 1
worker: #0      word: better    count: 1
worker: #0      word: Jude)     count: 1
worker: #0      word: to        count: 3
worker: #0      word: better)   count: 1
worker: #0      word: nah,      count: 25
worker: #0      word: someone   count: 1
worker: #0      word: (oh,      count: 1
worker: #0      word: you,      count: 1
worker: #0      word: and       count: 3
worker: #0      word: it's      count: 1
worker: #0      word: made      count: 1
worker: #0      word: know      count: 1
worker: #0      word: waiting   count: 1
worker: #0      word: afraid    count: 1
worker: #0      word: her       count: 2
worker: #0      word: Jude,     count: 4
worker: #1      word: song      count: 1
worker: #1      word: under     count: 1
worker: #1      word: You're    count: 1
worker: #1      word: Nah,      count: 5
worker: #1      word: You       count: 1
worker: #1      word: hey,      count: 1
worker: #1      word: begin     count: 2
worker: #1      word: perform   count: 1
worker: #1      word: skin      count: 1
worker: #1      word: And       count: 1
worker: #1      word: Hey,      count: 8
worker: #1      word: your      count: 1
worker: #1      word: go        count: 1
Worker #2 Final Input Lines 11~13:
Worker #3 Input Lines 11~15:
Worker #1 Final Input Lines 11~13:
Worker #0 Input Lines 11~15:
worker: #2      word: The       count: 1
worker: #1      word: Nah,      count: 5
worker: #1      word: And       count: 1
worker: #1      word: shoulders count: 1
worker: #1      word: fool      count: 1
worker: #1      word: upon      count: 1
worker: #1      word: your      count: 2
worker: #1      word: refrain   count: 1
worker: #1      word: movement  count: 1
worker: #1      word: Hey,      count: 5
worker: #0      word: on        count: 1
worker: #0      word: yeah,     count: 1
worker: #0      word: Jude,     count: 1
worker: #0      word: (ooh)     count: 1
worker: #0      word: For       count: 1
worker: #0      word: world     count: 1
worker: #0      word: yeah)     count: 1
worker: #0      word: nah,      count: 23
worker: #0      word: is        count: 1
```

```
worker: #0      word: pain,     count: 1
worker: #0      word: anytime   count: 1
worker: #0      word: well      count: 1
worker: #0      word: cool      count: 1
worker: #0      word: know      count: 1
worker: #0      word: it's      count: 1
worker: #2      word: nah       count: 17
worker: #2      word: (yeah,    count: 1
worker: #2      word: Who       count: 1
worker: #2      word: that      count: 1
worker: #2      word: Jude      count: 4
worker: #2      word: need      count: 1
worker: #2      word: carry     count: 1
worker: #2      word: yeah      count: 1
worker: #2      word: (Jude)    count: 1
worker: #2      word: it        count: 1
worker: #3      word: feel      count: 1
worker: #3      word: a count: 1
worker: #3      word: Don't     count: 1
worker: #3      word: plays     count: 1
worker: #3      word: the       count: 2
worker: #3      word: you       count: 3
worker: #3      word: shoulder  count: 1
Worker #3 Input Lines 16~20:
Worker #0 Final Input Lines 16~18:
worker: #2      word: Jude      count: 3
worker: #0      word: world     count: 1
worker: #0      word: By        count: 1
worker: #0      word: nah,      count: 13
worker: #0      word: colder    count: 1
worker: #0      word: making    count: 1
worker: #2      word: nah       count: 11
worker: #3      word: a count: 1
worker: #1      word: little    count: 1
worker: #1      word: Nah,      count: 3
worker: #1      word: Hey,      count: 3
worker: #1      word: his       count: 1
Worker #3 Final Input Lines 21~24:
worker: #1      word: Nah,      count: 2
worker: #3      word: [fade     count: 1
worker: #2      word: out]      count: 1
worker: #2      word: nah       count: 6
worker: #2      word: Jude      count: 2
worker: #1      word: Hey,      count: 2
worker: #0      word: nah,      count: 10
```