Parallel Architecture, Assignment 2

s1140740

The purpose of this report is to outline implementation details of a cache coherence simulator, argue its validity and discuss results of experiments that were performed by the simulator.

1 Implementation Details

All parts are completed fully. Both MSI and MESI protocols are supported. The cache coherence simulator is in a Python script `cache.py`.

1.1 Command Line Options

There are several command line options available to the Python script:

```
$ python3 cache.py [-h] [--lines LINES] [--words WORDS] [--mesi] tracefile
```

For example, the command to run a MESI protocol on the file `trace1.txt` with 1024 lines per cache and 4 words per line is:

```
$ python3 cache.py --lines 1024 --words 4 --mesi trace1.txt
```

1.2 Data Structures

Instruction Describes an operation that is executed (read or write), an identifier of the CPU executing the instruction, and an address of the word that is being used in the operation.

Line Describes a single line of cache. Contains the tag and index of the line.

Event Describes an event (hit or miss) that occurs after a certain CPU requests a line from a cache.

States are represented by a string with one capital letter and can be one of `M`, `E`, `S`, `I`, or possibly `None` if the line is not present in the cache. An operation is also a string with one capital letter can be either `R` (read) or `W` (write). Finally, an event can be either a `miss` or a `hit`.

1.3 Transitions

Transitions are represented as a map of triplets (current state, operation and event) to a new state. For example, the following entry `('S', 'W', 'miss'): 'M'` describes the following transition rule:

> If the line is in state `S(hared)` and a CPU performs a `W(rite)` and gets a `miss` from the cache, transition the line state to `M(odified)`.

Some transitions, could result in multiple states depending on states of other caches. For example, the following entry `('I', 'R', 'miss'): lambda *args: 'E' if exclusive(*args)else 'S'` describes the following transition rule:

> If the line is in state `I(nvalid)` and a CPU performs a `R(ead)` and gets a `miss` from the cache, transition to `E(xclusive)` if no other cache has access to the line or to `S(hared)` if the cache line is present in caches of other CPUs.

There are four such maps altogether: local and remote transitions for both protocols MSI and MESI.

1.4 Simulator Algorithm TODO!!!!!

1. After a line is read from the trace file it is parsed and converted into an `Instruction` object.
2. A `Line` object is created from the address. This gives us access to the tag and index associated with an address.
3. Next, a cache lookup is performed and results in an `Event` object.
4. Metrics are calculated and the cache states are logged into standard output if logging is enabled.
5. Finally state transitions are performed for the line in the local cache as well as other caches that contain the line. The final state is extracted from transition rules described in chapter 1.2.

1.5 Output Format

The script produces output in the following format:

```
<localCPU> - <opcode> - <address>:
```

Depending on whether it was a cache hit or miss this is followed by one of:

```
Tag <tag> not found in line <index> of local cache.
Tag <tag> found in state <localState> in line <index> of local cache.
```

Finally, if the line was found in remote caches, following message is appended:

```
Found in following remote caches [<remoteCPU>:<remoteState>, ...].
```

2 Validation of the Simulator

There are some unit tests which I produced while developing the simulator in `test.py`.

- The following example shows how a read to a line that is in state `Modified` in another cache, transitions the state of the line to state `Shared` for both caches (MSI protocol).

```
$ cat example1.txt
v
P1 W 01
P0 R 01
P1 R 01
$ python cache.py example1.txt
P1 - W - 00001: Tag 00 not found in line 0001 of local cache.
P0 - R - 00001: Tag 00 not found in line 0001 of local cache. Found in following
    remote caches P1: M.
P1 - R - 00001: Tag 00 found in state S in line 0001 of local cache. Found in
   following remote caches P0: S.
```

- The following example shows how a write to a line that is in state `Shared` in other caches, transitions the state of the line in local cache to state `Modified` and to state `Invalid` in remote caches (MSI protocol).

```
$ cat example2.txt
v
P1 R 02
P0 R 02
P2 R 02
P1 W 02
P0 W 02
$ python cache.py example2.txt
P1 - R - 00002: Tag 00 not found in line 0002 of local cache.
P0 - R - 00002: Tag 00 not found in line 0002 of local cache. Found in following
    remote caches P1: S.
P2 - R - 00002: Tag 00 not found in line 0002 of local cache. Found in following
    remote caches P0: S, P1: S.
P1 - W - 00002: Tag 00 found in state S in line 0002 of local cache. Found in
   following remote caches P0: S, P2: S.
P0 - W - 00002: Tag 00 found in state I in line 0002 of local cache. Found in
   following remote caches P1: M.
```

- The following example shows how a write to a line that is in state `Modified` in another cache, transitions the state of the line in local cache to state `Modified` and to state `Invalid` in the remote cache (MSI protocol).

```
$ cat example3.txt
v
P1 W 03
P0 W 03
P1 R 03
$ python cache.py example3.txt
P1 - W - 00003: Tag 00 not found in line 0003 of local cache.
P0 - W - 00003: Tag 00 not found in line 0003 of local cache. Found in following
     remote caches P1: M.
P1 - R - 00003: Tag 00 found in state I in line 0003 of local cache. Found in
    following remote caches P0: M.
```

- The following example shows that addresses are translated into a tag and an index correctly as well as that their command line options work (MSI protocol).

```
$ cat example5.txt
v
P0 R 0
P1 R 263
P2 R 512
P3 R 773
$ py cache.py example5.txt --words 8 --lines 256
P0 - R - 00000: Tag 00 not found in line 0000 of local cache.
P1 - R - 00263: Tag 01 not found in line 0001 of local cache.
P2 - R - 00512: Tag 02 not found in line 0000 of local cache.
P3 - R - 00773: Tag 03 not found in line 0001 of local cache.
```

- The following example shows how a read to a line that is in state `Exclusive` in another cache, transitions the state of the line to state `Shared` for both caches (MESI protocol).

```
$ cat example6.txt
v
P0 R 06
P1 R 06
P1 W 06
$ python cache.py example6.txt --mesi
P0 - R - 00006: Tag 00 not found in line 0006 of local cache.
P1 - R - 00006: Tag 00 not found in line 0006 of local cache. Found in following
     remote caches P0: E.
P1 - W - 00006: Tag 00 found in state S in line 0006 of local cache. Found in
    following remote caches P0: S.
```

3 Experiments

- sharing pattern of each cache-line/block
- the type of permissions required for each memory access
- miss rate for different types of accesses etc.
- Think of other statistics that will provide more information about the cache usage and the access patterns of the parallel application.

3.1 Vary the size of the cache and cache-line/block size to iterate over the above experiments. How do differing cache-line/block sizes and the number of cache- lines affect miss/hit rates and invalidations?

3.2 In terms of memory accesses, what is the distribution of accesses to private data (hits in state Modified, and also Exclusive for MESI), shared read-only data (hits in Shared, but never written to) and shared read-write data (hits in Shared, but modified).

3.3 If you manage to implement a MESI protocol, then you should also perform experiments to compare the two protocols and to identify the main advantages of it, if any.