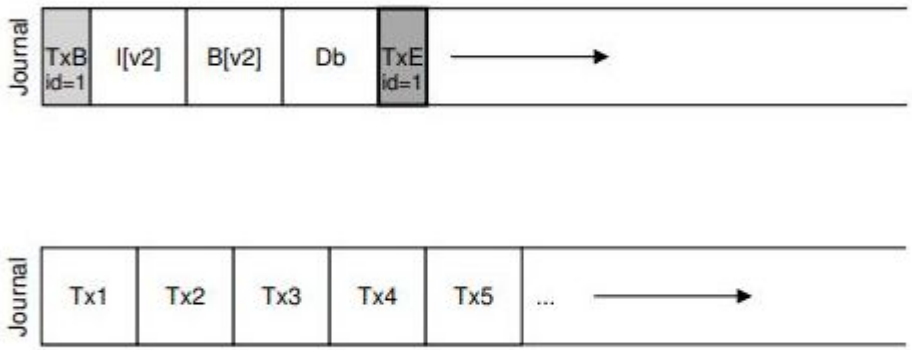
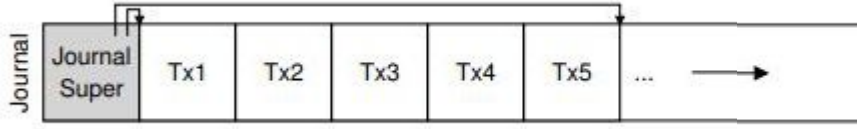
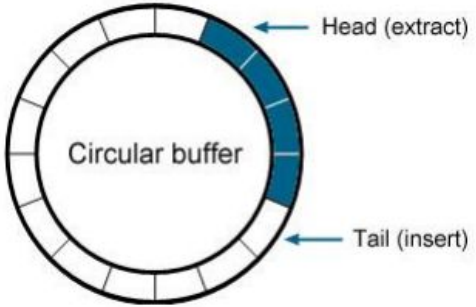
Journaling

Yijing Chen & Diyuan Dai


Journal
TxB
id=1
I[v2]
B[v2]
Db
TxE
id=1
Journal
Tx1
Tx2
Tx3
Tx4
Tx5
...

Inner Structure of Journal - Circular Buffer





In the demo, buffer size is fixed at 30
Which means once the Journal stores over 30 transaction
It will delete the previous transaction (at the very beginning) to free the space

Call Journal::Jwrite() instead of blockio::write_all()

make commit + call blockio::write_all()

Check commits in journal block

redo the commit

Thank you!

```c
int main(int argc, char** argv) {
    printf("\n");

    printf("opening data.txt: \n\n");
    int fd = open("/data/data.txt", 0);
    cp(fd,1);
    printf("\n");

    printf("writing to data.txt...\n\n");
     seek(fd, 4);
    write(fd, "Data has changed, do panic !",60 );

    seek(fd, 0);
    printf("data.txt after write(): \n\n");
    cp(fd,1);
    printf("\n\n");


    printf("Stress test, writing 50 times to data.txt\n");
    int curof =2;
    for(int i = 0; i<50; i++) {
        seek(fd, curof);
        write(fd, "was", 3);
        curof += 3;
    }
    printf("\n");
    seek(fd, 0);
    cp(fd,1);
    printf("\n");
    visualizeJournal();
    shutdown();
    return 0;
}
```

Demo code:

--Test case

Writing 50 times which will go over the buffer size which is 30.
It will delete the first 20 transactions to save space

```
ssize_t write(void* buf, size_t size)  {
  // Debug::printf("))))openFile wirte called\n");
   using namespace gheith;
   // call the journal to buffer the content

   char* bufferContent = root_fs->jsb->Jwrite(node->number, offset, size, (char*)buf);

   // checkpointing the writing
   // redo the content
   auto lastTX = root_fs->jsb->getlastTX();

   auto tgtInumFromJournal = lastTX->inum;
   auto tgtNode = root_fs->get_node(tgtInumFromJournal);
   auto tgtSize = lastTX->tgtsize;
   auto tgtoffs = lastTX->tgtoffs;
   auto cunt = tgtNode->write(tgtoffs,tgtSize,(char*)bufferContent);
  // auto cunt = node ->write(offset, size, (char*)buf);
   offset += cunt;
   return cunt;
}
```

Substitution of write:

Buffered into Journal first
Then,
Do the checkpointing

```
343
344 uint32_t txCommit(uint32_t inumber,uint32_t offs, uint32_t size, char* buffer){
345
346     Transaction* newTXwithID = txQueue->newTx();
347     txQueue->addTx(newTXwithID);
348      Debug::printf("\nCalling Journal:___");
349
350     // mark the txstart
351     newTXwithID->markStart();
352     Debug::printf("TXStart marked___");
353     // set node and bitmap content
354     newTXwithID->setNode(inumber);
355     // write in the buffer inside of transction
356       newTXwithID->tgtoffs = offs;
357     newTXwithID->tgtsize = size;
358     newTXwithID->writeinTx(buffer);
359
360     // Safely get buffered into the Journal!!!
361     // mark the txend
362       Debug::printf("Og datas and dest offs buffered___");
363     newTXwithID->markEnd();
364      Debug::printf("TXEnd marked___\n");
365
366     // one tranction completed
367
368     return newTXwithID->txID;
369 }// return the newest TXID
370
```

While making a commit to Journal:

Mark TXstart first
Then the original data and target destination
Finally TXend to mark complete