Recitation 7

Zeinab Khalifa

October 8th, 2020

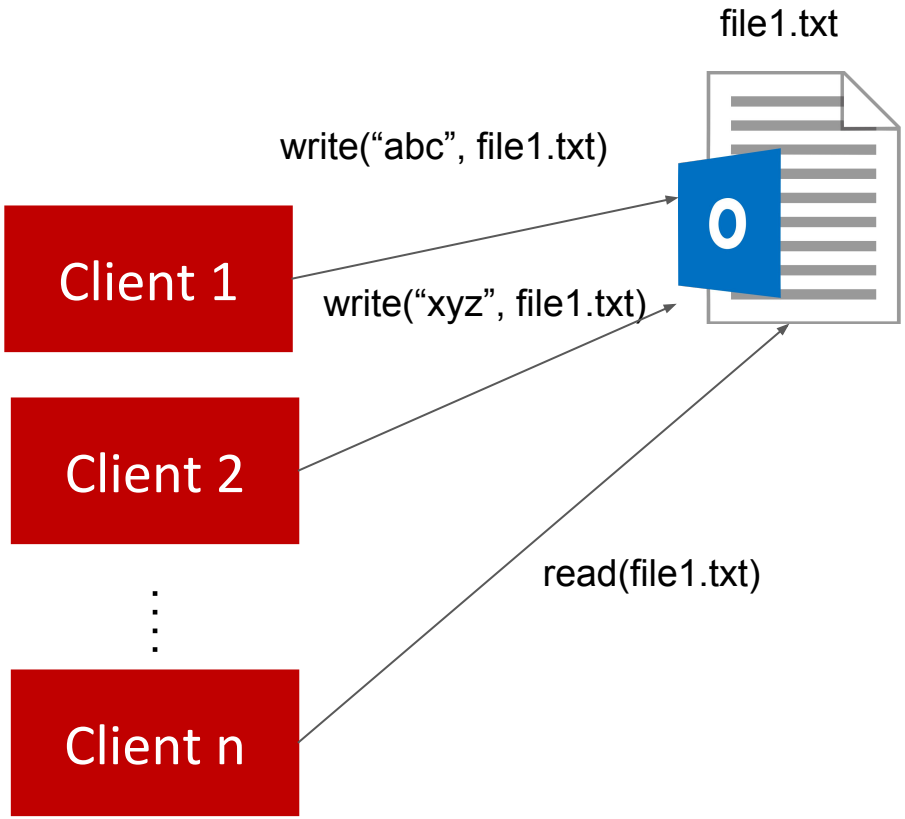
Carnegie Mellon University Qatar

The bank use case example is from 6.189 IAP 2007 MIT concurrent programming lecture

Announcements

- PS3 is due on Monday, October 12, 2020
- P2 will be out today and is due on October 28, 2020

What did we do last time?



- Synchronization
- Load-balancing
- Consistency

Project 2 Objectives

Synchronization

Readers

Shared Lock

Can multiple readers read the file simultaneously?

Can multiple readers read the file while someone is writing to it?

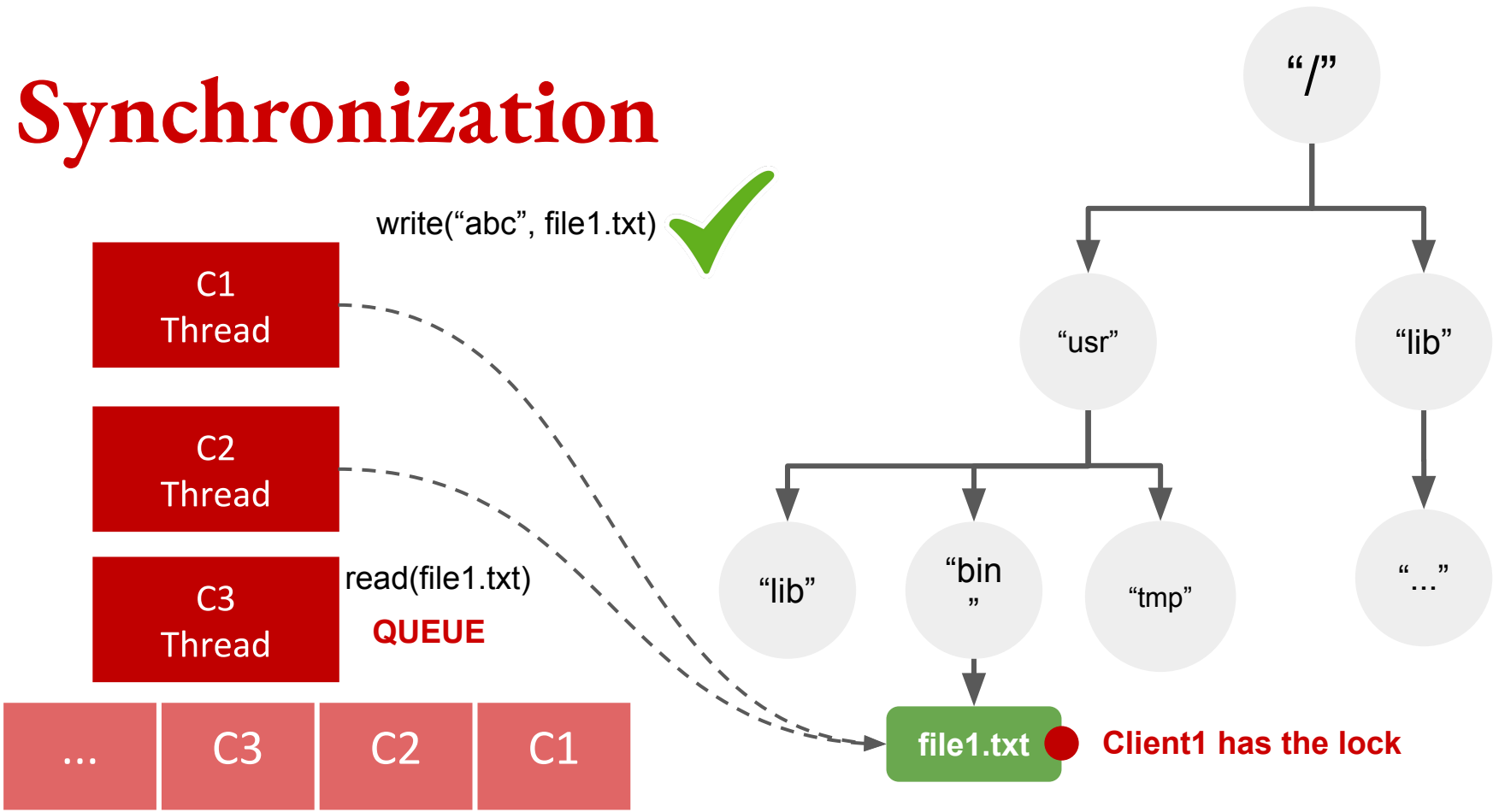
Writers

Exclusive Lock

Can multiple people write to the same file?

Can multiple people write to a file while someone is reading it?

Synchronization

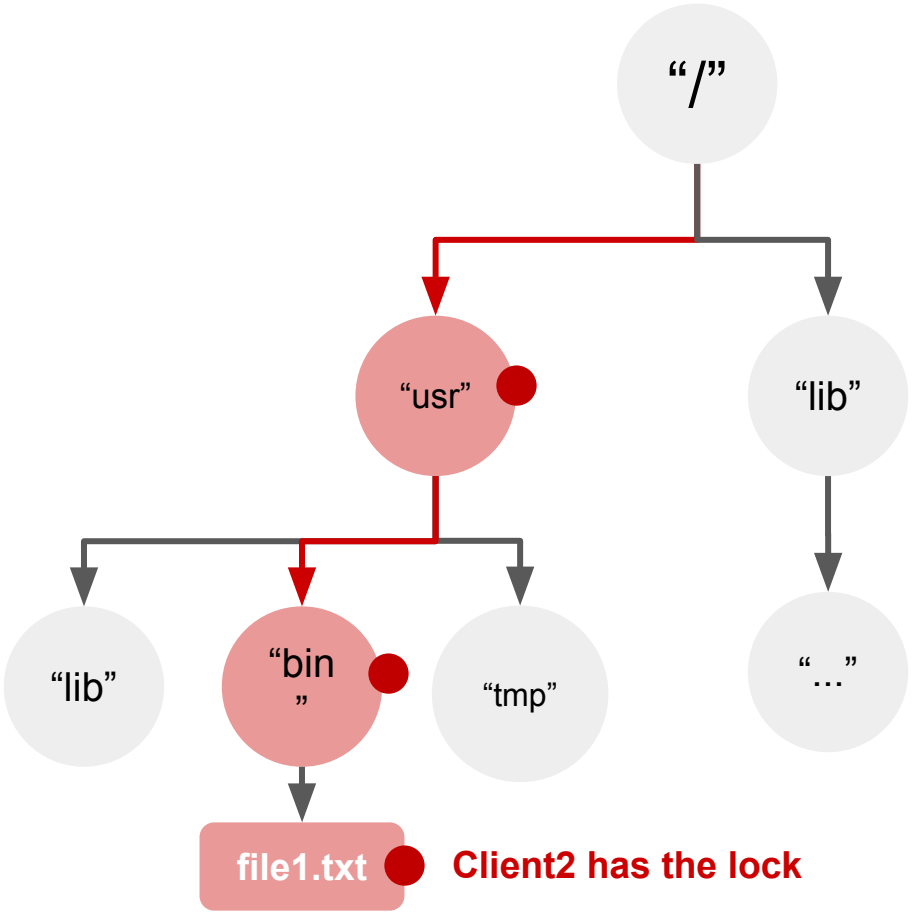


Synchronization

How are locks implemented?

```
Node {
        synchronized obtain_lock() {
                ...
                wait();
        }

        synchronized release_lock() {
                ...
                notifyAll();
        }
}
```

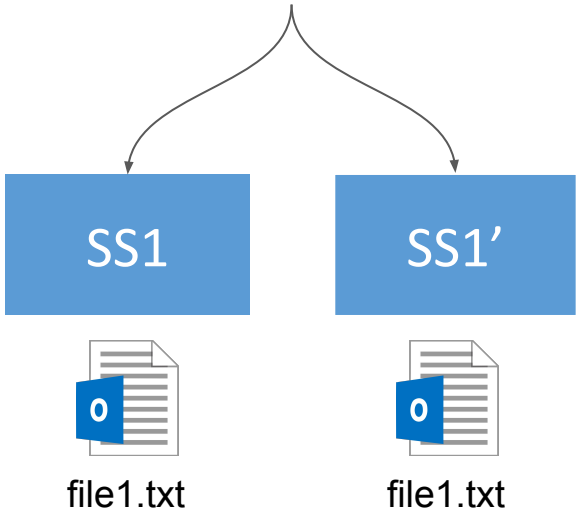


Load Balancing

How are we scaling?

HOT FILES

Frequently Accessed



num_replicas = ALPHA * num_requesters

num_replicas = min (ALPHA * num_requesters , REPLICA_UPPER_BOUND)

num_requesters_coarse = {N | N >= num_requesters & a m ul ti pl e o f 20}

num_replicas = min (ALPHA * num_requesters_coarse , REPLICA_UPPER_BOUND)

Synchronization - Bank Use Case

Synchronization - Bank Use Case

We want to implement the code for a bank with multiple ATMs.

```
import java.util.*;

public class Account {
	String id;
	String password;
	int balance;

	Account(String id, String password, String balance) {
		this.id = id;
		this.password = password;
		this.balance = balance;
	}

	boolean is_password(String password) {
		return password == this.password;
	}

	int getbal() {
		return balance;
	}

	void post(int v) {
		balance = balance + v;
	}

}
```

```java
import java.util.*;

public class Account {
    String id;
    String password;
    int balance;

    Account(String id, String password, String balance) {
        this.id = id;
        this.password = password;
        this.balance = balance;
    }

    boolean is_password(String password) {
        return password == this.password;
    }

    int getbal() {
        return balance;
    }

    void post(int v) {
        balance = balance + v;
    }

}
```

```java
import java.util.*;

public class Bank {
    HashMap<String, Account> accounts;
    static Bank theBank = null;

    private Bank() {
        accounts = new HashMap<String, Account>();
    }

    public static Bank getbank() {
        if (theBank == null)
            theBank = new Bank();
        return theBank;
    }

    public Account get(String ID) {
        return accounts.get(ID);
    }
    ...
}
```

```
import java.util.*;
import java.io.*;

public class ATM {
      static Bank bnk;
      PrintStream out;
      BufferedReader in;

      ATM(PrintStream out, BufferedReader in) {
         this.out = out;
         this.in = in;
      }

       public static void main(String[] args) {
          bnk = Bank.getbank();
         BufferedReader stdin = new BufferedReader
              (new InputStreamReader(System.in));
         ATM atm = new ATM(System.out, stdin);
         atm.run();
       }
```

```
import java.util.*;
import java.io.*;

public class ATM {
      static Bank bnk;
      PrintStream out;
      BufferedReader in;

      ATM(PrintStream out, BufferedReader in) {
         this.out = out;
         this.in = in;
      }

       public static void main(String[] args) {
          bnk = Bank.getbank();
         BufferedReader stdin = new BufferedReader
              (new InputStreamReader(System.in));
         ATM atm = new ATM(System.out, stdin);
         atm.run();
       }
```

```
public void run() {
     while(true) {
        try {
           out.print( "Account ID > " );
           String id = in.readLine();
           String acc = bnk.get(id);
           if (acc == null) throw new Exception();
           out.print( "Password > " );
           String pass = in.readLine();
           if (!acc.is_password(pass))
              throw new Exception();
           out.print("your balance is " + acc.getbal());
           out.print( "Deposit or withdraw amount > " );
           int val = in.read();
           if (acc.getbal() + val > 0)
              acc.post(val);
          else
              throw new Exception();
           out.print("your balance is " + acc.getbal());
        } catch(Exception e) {
           out.println( "Invalid input, restart" );
        }
     }
  }
}
```

How can we run multiple ATMs?

```
import java.util.*;
import java.io.*;

public class ATM {

    static Bank bnk;
    PrintStream out;
    BufferedReader in;


    ATM(PrintStream out, BufferedReader in) {
        this.out = out;
        this.in = in;
    }


     public static void main(String[] args) {
         bnk = Bank.getbank();
         BufferedReader stdin = new BufferedReader
             (new InputStreamReader(System.in));
         ATM atm = new ATM(System.out, stdin);
         atm.run();
     }
```

```
public void run() {
        while(true) {
            try {
                out.print( "Account ID > " );
                String id = in.readLine();
                String acc = bnk.get(id);
                if (acc == null) throw new Exception();
                out.print( "Password > " );
                String pass = in.readLine();
                if (!acc.is_password(pass))
                   throw new Exception();
                out.print("your balance is "  +  acc.getbal());
                out.print( "Deposit or withdraw amount > " );
                int val = in.read();
                if (acc.getbal() + val > 0)
                    acc.post(val);
                else
                   throw new Exception();
                out.print("your balance is "  +  acc.getbal());
            } catch(Exception e) {
                out.println( "Invalid input, restart" );
            }
        }
    }
}
```

```
import java.util.*;
import java.io.*;

public class ATMs extends Thread {
      static final int numATMs = 4;
      static Bank bnk;
      PrintStream out;
      BufferedReader in;
      int atmnum;

      ATMs(int num, PrintStream out, BufferedReader in) {
            this.out = out;
            this.in = in;
            this.atmnum = num;
      }

       public static void main(String[] args) {
             bnk = Bank.getbank();
             ATMs atm[] = new ATMs[numATMs];
             for(int i=0; i<numATMs; i++){
                 atm[i] = new ATMs(i, outdevice(i), indevice(i));
                 atm[i].start();
             }
       }
```

```
public void run() {
      while(true) {
          try {
              out.print("Account ID > " );
              String id = in.readLine();
              String acc = bnk.get(id);
              if (acc == null) throw new Exception();
              out.print( "Password > " );
              String pass = in.readLine();
              if (!acc.is_password(pass))
                 throw new Exception();
              out.print("your balance is "  +  acc.getbal());
              out.print( "Deposit or withdraw amount > " );
              int val = in.read();
              if (acc.getbal() + val > 0)
                  acc.post(val);
             else
                 throw new Exception();
              out.print("your balance is "  +  acc.getbal());
          } catch(Exception e) {
              out.println( "Invalid input, restart" );
          }
      }
  }
}
```

balance

ATM 1

100

```
out.print("your balance is " + acc.getbal());
Your account balance is 100

out.print("Deposit or withdraw amount > ");
Deposit or Withdraw amount >

     -90
int val = in.read();
```

100

```
if (acc.getbal() + val > 0)
```

100

```
 acc.post(val);
```

10

10

```
out.print("your balance is " + acc.getbal());
Your account balance is 10
```

balance

100

100

100

10

10

ATM 1

```
out.print("your balance is " + acc.getbal());
Your account balance is 100

out.print("Deposit or withdraw amount > ");
Deposit or Withdraw amount >

    -90
int val = in.read();

if (acc.getbal() + val > 0)


 acc.post(val);


out.print("your balance is " + acc.getbal());
Your account balance is 10
```

ATM 2

```
out.print("your balance is " + acc.getbal());
Your account balance is 100

out.print("Deposit or withdraw amount > ");
Deposit or Withdraw amount >

    -90

int val = in.read();

if (acc.getbal() + val > 0)

  acc.post(val);


out.print("your balance is " + acc.getbal());
Your account balance is 10
```

```
synchronized int getbal() {
        return balance;
    }

synchronized void post(int v) {
        balance = balance + v;
    }
```

balance	ATM 1	ATM 2
100		int val = in.read();
	int val = in.read();	
100		if (acc.getbal() + val > 0)
100	if (acc.getbal() + val > 0)	
100		acc.post(val);
10	acc.post(val);	
-80		
		out.print("your balance is " + acc.getbal());
		Your account balance is -80
	out.print("your balance is " + acc.getbal());	
	Your account balance is -80	

Negative Bank Balance!

```
synchronized (acc) {
        if (acc.getbal() + val > 0)
                acc.post(val);
        else
                throw new Exception();
        out.print("your balance is " + acc.getbal());
}
```

```
import java.util.*;
import java.io.*;

public class ATMs extends Thread {
      static final int numATMs = 1;
      static Bank bnk;
      PrintStream out;
      BufferedReader in;
      int atmnum;

      ATMs(int num, PrintStream out, BufferedReader in) {
            this.out = out;
            this.in = in;
            this.atmnum = num;
      }

       public static void main(String[] args) {
            bnk = Bank.getbank();
            ATMs atm[] = new ATMs[numATMs];
            for(int i=0; i<numATMs; i++){
                atm[i] = new ATMs(i, outdevice(i), indevice(i));
                atm[i].start();
            }
       }
```

```
public void run() {
      while(true) {
          try {
              out.print("Account ID > " );
              String id = in.readLine();
              String acc = bnk.get(id);
              if (acc == null) throw new Exception();
              out.print("Password > " );
              String pass = in.readLine();
              if (!acc.is_password(pass))
                 throw new Exception();
              out.print("your balance is " + acc.getbal());
              out.print("Deposit or withdraw amount > " );
              int val = in.read();
              synchronized (acc) {
                 if (acc.getbal() + val > 0)
                   acc.post(val);
                 else
                   throw new Exception();
                 out.print("your balance is " + acc.getbal());
              }
          } catch(Exception e) {
              out.println("Invalid input, restart" );
          }
      }
  }
}
```

balance	ATM 1	ATM 2
100		out.print("your balance is " + acc.getbal());
100	out.print("your balance is " + acc.getbal());	Your account balance is 100
	Your account balance is 100	
		out.print("Deposit or withdraw amount > ");
	out.print("Deposit or withdraw amount > ");	Deposit or Withdraw amount >
	Deposit or Withdraw amount >	
	-90	-90
	int val = in.read();	int val = in.read();
100	synchronized(acc)	
100	if (acc.getbal() + val > 0)	synchronized(acc)
10	acc.post(val);	
	out.print("your balance is " + acc.getbal());	
10	Your account balance is 10	
10		if (acc.getbal() + val > 0)
		throw new Exception()

Balance shows 100, but couldn't withdraw!

```
import java.util.*;
import java.io.*;

public class ATMs extends Thread {
      static final int numATMs = 1;
      static Bank bnk;
      PrintStream out;
      BufferedReader in;
      int atmnum;

      ATMs(int num, PrintStream out, BufferedReader in) {
            this.out = out;
            this.in = in;
            this.atmnum = num;
      }

       public static void main(String[] args) {
            bnk = Bank.getbank();
            ATMs atm[] = new ATMs[numATMs];
            for(int i=0; i<numATMs; i++){
                atm[i] = new ATMs(i, outdevice(i), indevice(i));
                atm[i].start();
            }
       }
```

```
public void run() {
      while(true) {
         try {
            out.print("Account ID > " );
            String id = in.readLine();
            String acc = bnk.get(id);
            if (acc == null) throw new Exception();
            out.print("Password > " );
            String pass = in.readLine();
            if (!acc.is_password(pass))
               throw new Exception();
            synchronized (acc) {
               out.print("your balance is " + acc.getbal());
               out.print("Deposit or withdraw amount > " );
               int val = in.read();
               if (acc.getbal() + val > 0)
                 acc.post(val);
               else
                 throw new Exception();
               out.print("your balance is " + acc.getbal());
            }
         } catch(Exception e) {
            out.println("Invalid input, restart" );
         }
      }
   }
}
```

ATM 1

```
Account ID >
```

ben

```
Password >
```

6189cell

```
synchronized(acc)
out.print("your balance is " + acc.getbal());
Your account balance is 100

out.print("Deposit or withdraw amount > ");
Deposit or Withdraw amount >
```

-90

ATM 2

```
Account ID >
```

ben

```
Password >
```

6189cell

```
synchronized(acc)
```

```
public boolean transfer(Account from, Account to, int val) {
        synchronized(from) {
            if (from.getbal() > val)
                from.post(-val);
            else
                throw new Exception();
            synchronized(to) {
                to.post(val);
            }
        }
    }
```

```
public boolean transfer(Account from, Account to, int val) {
        synchronized(from) {
            if (from.getbal() > val)
                from.post(-val);
            else
                throw new Exception();
            synchronized(to) {
                to.post(val);
            }
        }
    }
```

Allyssa wants to transfer $10 to Ben's account
While Ben wants to also transfer $20 to Allyssa's account

Allyssa→Ben **Ben→Allysa**

Allyssa→Ben

```
synchronized(from)
if (from.getbal() > val)
from.post(-val);

    synchronized(to)
```

Waiting for Ben's account to be released to perform

Ben→Allyssa

```
synchronized(from)
if (from.getbal() > val)
from.post(-val);

    synchronized(to)
```

Waiting for Allyssa's account to be released to perform

DEADLOCKED!

```
public class Account {
    String id;
    String password;
    int balance;
    static int count;
    public int rank;

    Account(String id,
            String password,
            String balance) {
        this.id = id;
        this.password = password;
        this.balance = balance;
        rank = count++;
    }

    …

}
```

```
…
public boolean transfer(Account from,
                        Account to,
                        int val) {

    Account first = (from.rank > to.rank)?from:to;
    Account second = (from.rank > to.rank)?to:from;
        synchronized(first) {
            synchronized(second) {
                if (from.getbal() > val)
                    from.post(-val);
                else
                    throw new Exception();
                to.post(val);
            }
        }
}
```

Let’s explore the Debugger!

References

- https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf
- The bank use case code and slides are from 6.189 IAP 2007 MIT concurrent programming lecture.