To design a distributed application using MapReduce in Java for processing a log file and finding the users who have logged in for the maximum period on the system, you can follow these steps:

Step 1: Input Log File

Create a text file containing log entries, where each line represents a single log entry with user information and timestamp.

Example log file (log.txt):

```
user1 2023-05-01 08:00:00
user2 2023-05-01 09:00:00
user1 2023-05-01 09:30:00
user2 2023-05-01 10:00:00
user1 2023-05-02 08:00:00
user2 2023-05-02 09:00:00
user1 2023-05-02 09:30:00
user2 2023-05-02 10:00:00
user1 2023-05-03 08:00:00
user2 2023-05-03 09:00:00
user1 2023-05-03 09:30:00
```

Step 2: Set up Hadoop

Set up a Hadoop cluster or use Hadoop in pseudo-distributed mode on your local machine. Ensure that you have Java and Hadoop installed and configured correctly.

Step 3: Create MapReduce Java Project

Create a new Java project in your preferred IDE (e.g., Eclipse or IntelliJ).

Step 4: Implement Mapper and Reducer

Create a new Java class, `LogAnalyzerMapper`, that extends `Mapper<LongWritable, Text, Text, LongWritable>`. Implement the `map` function to extract user information and timestamp from each log entry.

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class LogAnalyzerMapper extends Mapper<LongWritable, Text, Text, LongWritable> {
    private final Text user = new Text();
    private final LongWritable duration = new LongWritable();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] tokens = line.split(" ");

        if (tokens.length >= 2) {
            user.set(tokens[0]);
            duration.set(Long.parseLong(tokens[1]));
            context.write(user, duration);
        }
    }
}
```

Create another Java class, `LogAnalyzerReducer`, that extends `Reducer<Text, LongWritable, Text, LongWritable>`. Implement the `reduce` function to calculate the total duration for each user.

```java
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class LogAnalyzerReducer extends Reducer<Text, LongWritable, Text, LongWritable> {
    private final LongWritable totalDuration = new LongWritable();

    @Override
    protected void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException, InterruptedException {
        long sum = 0;
        for (LongWritable duration : values) {
            sum += duration.get();
        }

        totalDuration.set(sum);
        context.write(key, totalDuration);
    }
}
```

Step 5: Configure and Run MapReduce Job

Create a new Java class, `LogAnalyzerJob`, to configure and run the MapReduce job.

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;


import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class LogAnalyzerJob {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Log Analyzer");

        job.setJarByClass(LogAnalyzerJob.class);
        job.setMapperClass(LogAnalyzerMapper.class);
        job.setReducerClass(LogAnalyzerReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

}

```

Step 6: Build and Package the Project

Build the Java project and package it as a JAR file.

Step 7: Upload Log File to Hadoop

Upload the log file (`log.txt`) to the Hadoop Distributed File System (HDFS) or make it available in the Hadoop input directory.

Step 8: Execute the MapReduce Job

Execute the MapReduce job by running the following command:

```

hadoop jar log-analyzer.jar LogAnalyzerJob <input_path> <output_path>

```

Replace `<input_path>` with the path to the input log file in HDFS, and `<output_path>` with the desired output directory path in HDFS.

Step 9: View the Results

After the MapReduce job completes, you can view the output file containing the users and their total durations in the specified output directory.

To find the user(s) who have logged in for the maximum period on the system, you can further process the output using another program or script to identify the user(s) with the highest duration.

Note: This is a basic example that assumes the log entries have a user and timestamp format. You may need to modify the code accordingly if your log file structure is different.
```