**Student:** Hong Quan Doan

**Student ID:** 986956

**Day 16, 17 – Mar 15, 16**

**Environment**

MacBook, Core i5, 4 CPU, 8GB

**Numerical Integration**

**Test results:**

Range: 2,000,000,000

Non-parallel stream ===

Result: 3.141593

Elapsed time: 18,529

Parallel stream ===

Result: 3.141593

Elapsed time: 9,109

Speedup: 2.0341 ms

**Source code:**

**public** **static** **void** main(String[] args) {

**long** start, regTime, parTime;

**int** a = 0, b = 2, n = 2000000000;

**double** w = (**double**)(b - a)/n;

System.***out***.println(String.*format*("Range: %,3d", n));

System.***out***.println("Non-parallel stream ===");

start = System.*currentTimeMillis*();

**double** result = IntStream.*range*(1, n).asDoubleStream().map(i -> *f*(a + i \* w)).sum();

**double** ans = w \* (result + (*f*(a)+*f*(b))/2.0);

System.***out***.println(String.*format*("Result: %.6f", ans));

regTime = System.*currentTimeMillis*() - start;

System.***out***.println(String.*format*("Elapsed time: %,3d", regTime));

System.***out***.println("\nParallel stream ===");

start = System.*currentTimeMillis*();

result = IntStream.*range*(1, n).asDoubleStream().parallel().map(i -> *f*(a + i \* w)).sum();

ans = w \* (result + (*f*(a)+*f*(b))/2.0);

System.***out***.println(String.*format*("Result: %.6f", ans));

parTime = System.*currentTimeMillis*() - start;

System.***out***.println(String.*format*("Elapsed time: %,3d", parTime));

System.***out***.println(String.*format*("Speedup: %.4f ms", (**double**)regTime / parTime));

}

**Student Grades project**

Total students: 1,000,000

Generating done!

Using regular stream

Average GPA of all students: 3.000

Lowest GPA of all students: 2.057

Highest GPA of all students: 3.950

Average GPA of Dept\_3: 3.001

Average GPA of Dept\_2: 3.000

Average GPA of Dept\_4: 3.000

Average GPA of Dept\_1: 3.000

Average GPA of Course\_9 is: 3.000

Average GPA of Course\_8 is: 3.001

Average GPA of Course\_5 is: 2.999

Average GPA of Course\_4 is: 3.001

Average GPA of Course\_7 is: 2.999

Average GPA of Course\_6 is: 2.999

Average GPA of Course\_1 is: 3.000

Average GPA of Course\_3 is: 3.001

Average GPA of Course\_2 is: 3.000

Time elapsed: 3202 ms

Using parallel stream

Average GPA of all students: 3.000

Lowest GPA of all students: 2.057

Highest GPA of all students: 3.950

Average GPA of Dept\_3: 3.001

Average GPA of Dept\_1: 3.000

Average GPA of Dept\_2: 3.000

Average GPA of Dept\_4: 3.000

Average GPA of Course\_3 is: 3.001

Average GPA of Course\_5 is: 2.999

Average GPA of Course\_9 is: 3.000

Average GPA of Course\_7 is: 2.999

Average GPA of Course\_2 is: 3.000

Average GPA of Course\_8 is: 3.001

Average GPA of Course\_4 is: 3.001

Average GPA of Course\_6 is: 2.999

Average GPA of Course\_1 is: 3.000

Time elapsed: 2614 ms

Speedup: 1.2249 ms

**Source code**

**public** **static** **void** main(String[] args) {

**int** numStudents = 1000000;

**int** numCourses = 5;

**int** numDepts = 5;

**long** regTime, parTime;

System.***out***.println(String.*format*("Total students: %,3d", numStudents));

List<Student> studentList = **new** ArrayList<Student>();

**for** (**int** i = 0; i < numStudents; i++) {

String sid = String.*format*("%06d", i);

String sname = "TestStudent" + sid;

String dept = "Dept\_" + ThreadLocalRandom.*current*().nextInt(1, numDepts);

Student s = **new** Student(sid, sname, dept);

**for** (**int** j = 0; j < numCourses; j++) {

s.addStudentGrade(**new** StudentGrade(sid,

"Course\_" + ThreadLocalRandom.*current*().nextInt(1, numCourses \* 2), LocalDate.*now*(),

ThreadLocalRandom.*current*().nextInt(2, 4), ThreadLocalRandom.*current*().nextDouble(2.0, 4.0)));

}

studentList.add(s);

}

System.***out***.println("Generating done!");

System.***out***.println("\nUsing regular stream");

**long** start = System.*currentTimeMillis*();

**double** averageGpa = studentList.stream().mapToDouble(g -> g.computeGpa()).average().getAsDouble();

System.***out***.println(String.*format*("Average GPA of all students: %.3f", averageGpa));

**double** lowestGpa = studentList.stream().mapToDouble(g -> g.computeGpa()).min().getAsDouble();

System.***out***.println(String.*format*("Lowest GPA of all students: %.3f", lowestGpa));

**double** highestGpa = studentList.stream().mapToDouble(g -> g.computeGpa()).max().getAsDouble();

System.***out***.println(String.*format*("Highest GPA of all students: %.3f", highestGpa));

studentList.stream().collect(*groupingBy*(Student::getDepartmentId, *averagingDouble*(s -> s.computeGpa())))

.entrySet().stream().forEach(s -> System.***out***

.println("Average GPA of " + s.getKey() + String.*format*(": %.3f", s.getValue())));

Map<String, List<StudentGrade>> filterList = studentList.stream().flatMap(s -> s.getStudentGrade().stream())

.collect(*groupingBy*(StudentGrade::getCourseNumber));

filterList.entrySet().stream()

.forEach(s -> System.***out***.println(String.*format*("Average GPA of %s is: %.3f", s.getKey(),

s.getValue().stream().mapToDouble(m -> m.getCredits() \* m.getGrade()).sum()

/ s.getValue().stream().mapToInt(m -> m.getCredits()).sum())));

**long** end = System.*currentTimeMillis*();

regTime = end - start;

System.***out***.println(String.*format*("Time elapsed: %d ms", regTime));

System.***out***.println("\nUsing parallel stream");

start = System.*currentTimeMillis*();

System.***out***.println(String.*format*("Average GPA of all students: %.3f",

studentList.parallelStream().mapToDouble(g -> g.computeGpa()).average().getAsDouble()));

System.***out***.println(String.*format*("Lowest GPA of all students: %.3f",

studentList.parallelStream().mapToDouble(g -> g.computeGpa()).min().getAsDouble()));

System.***out***.println(String.*format*("Highest GPA of all students: %.3f",

studentList.parallelStream().mapToDouble(g -> g.computeGpa()).max().getAsDouble()));

studentList.parallelStream().collect(*groupingBy*(Student::getDepartmentId, *averagingDouble*(s -> s.computeGpa())))

.entrySet().parallelStream().forEach(s -> System.***out***

.println("Average GPA of " + s.getKey() + String.*format*(": %.3f", s.getValue())));

Map<String, List<StudentGrade>> parfilterList = studentList.parallelStream()

.flatMap(s -> s.getStudentGrade().parallelStream()).collect(*groupingBy*(StudentGrade::getCourseNumber));

parfilterList.entrySet().parallelStream()

.forEach(s -> System.***out***.println(String.*format*("Average GPA of %s is: %.3f", s.getKey(),

s.getValue().parallelStream().mapToDouble(m -> m.getCredits() \* m.getGrade()).sum()

/ s.getValue().parallelStream().mapToInt(m -> m.getCredits()).sum())));

end = System.*currentTimeMillis*();

parTime = end - start;

System.***out***.println(String.*format*("Time elapsed: %d ms", parTime));

System.***out***.println(String.*format*("Speedup: %.4f ms", (**double**)regTime / parTime));

}