Parallelise Kiwiplan Framework

Prabhjot Singh

October 20, 2010

Agenda

Background

- Kiwiplan displays the information to the user in the form of 2D tables.

Background

- Kiwiplan displays the information to the user in the form of 2D tables.

- Users can customize the table according to their requirements.

Background

- Kiwiplan displays the information to the user in the form of 2D tables.
- Users can customize the table according to their requirements.
- Some of the common operations - Sorting, filter out certain rows, group the table.

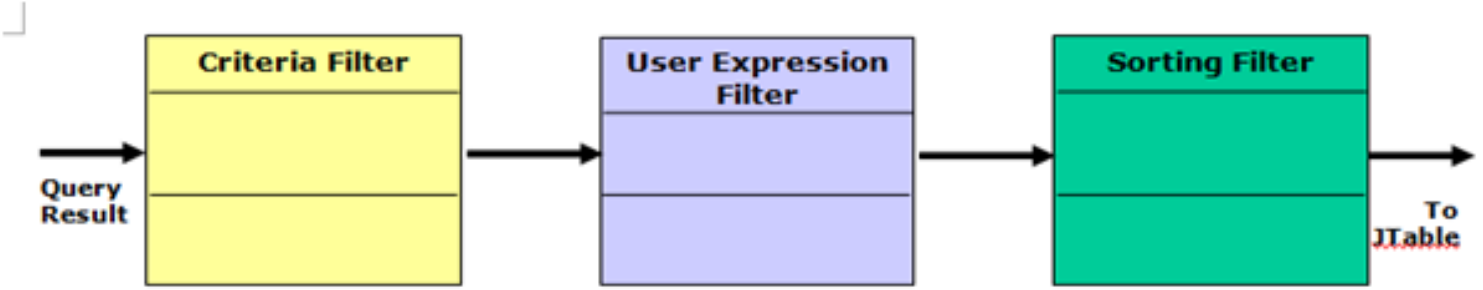
Background

Test table

Average GPA	Supervisor	Course
18	X0zfbnJ	▸ 18
47	NliNkK	▸ 47
107	rLMsGUH	▸ 107
39	FWLvnKe	▸ 39
64	gFCMm	▸ 64
92	wQlusWz	▸ 92
21	XenqQ	▸ 21
27	hBnvr	▸ 27
101	XmYfWe	▸ 101
31	ubVAhay	▸ 31

Background

- When the user perform certain operation on the interface, the request is passed on to the pipeline.



Project Goal

- Each of the filter algorithm is being written as a single threaded program.

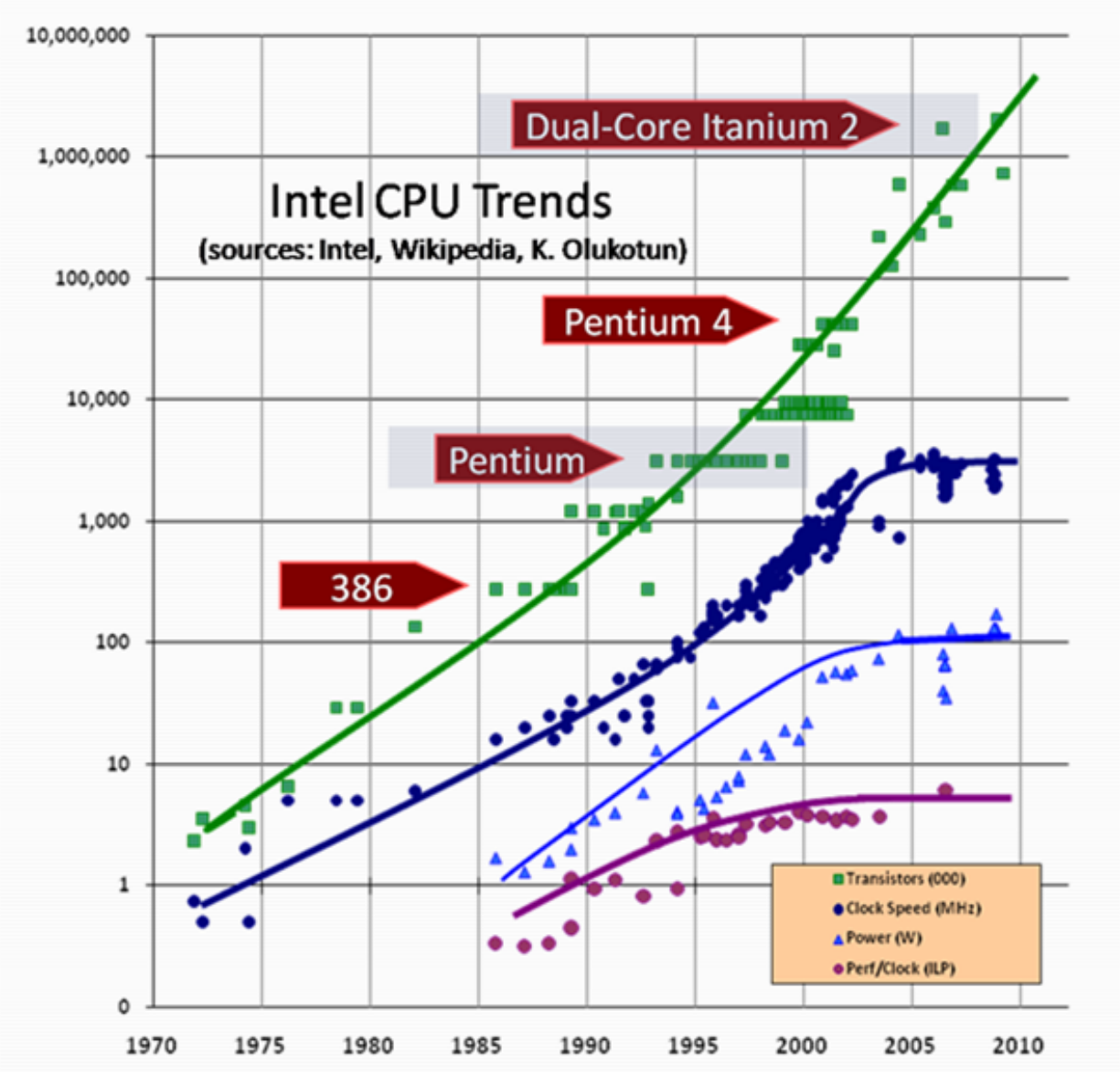
Project Goal

- Each of the filter algorithm is being written as a single threaded program.
- Goal is to make the algorithm to use multi-cores of the computer, if possible.

Project Goal

- Each of the filter algorithm is being written as a single threaded program.
- Goal is to make the algorithm to use multi-cores of the computer, if possible.
- Focuses mainly on the new Fork/Join framework.

Free Lunch is Over



Issue with Threads

- Let's consider one simple example.

Issue with Threads

- Let's consider one simple example.

- int value = 0;
  public int getNext() {
    return ++value;
  }

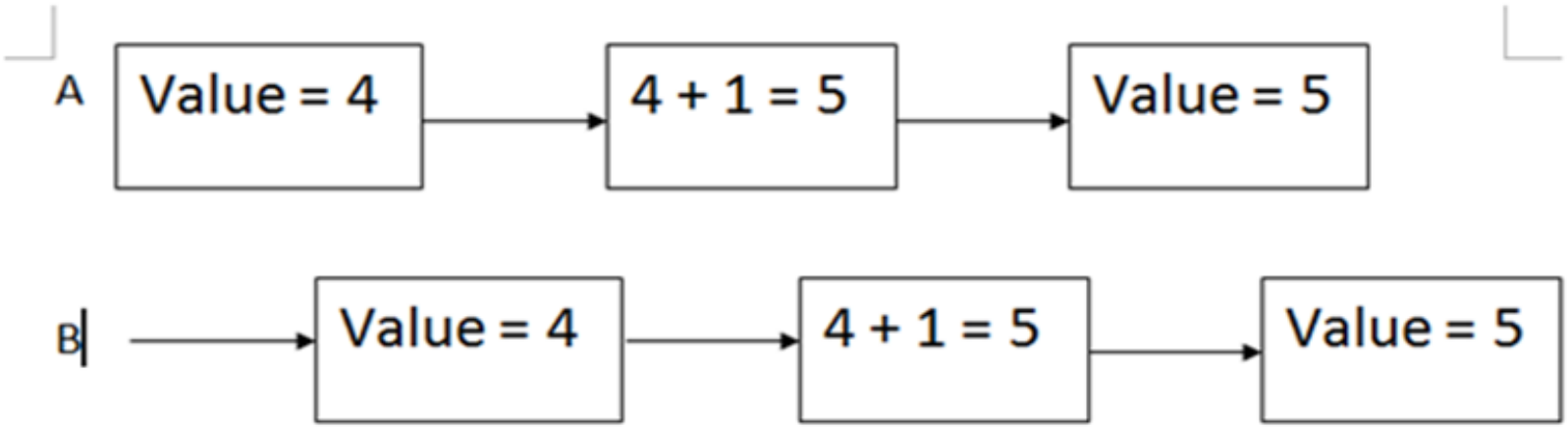
Issue with Threads

- Let's consider one simple example.

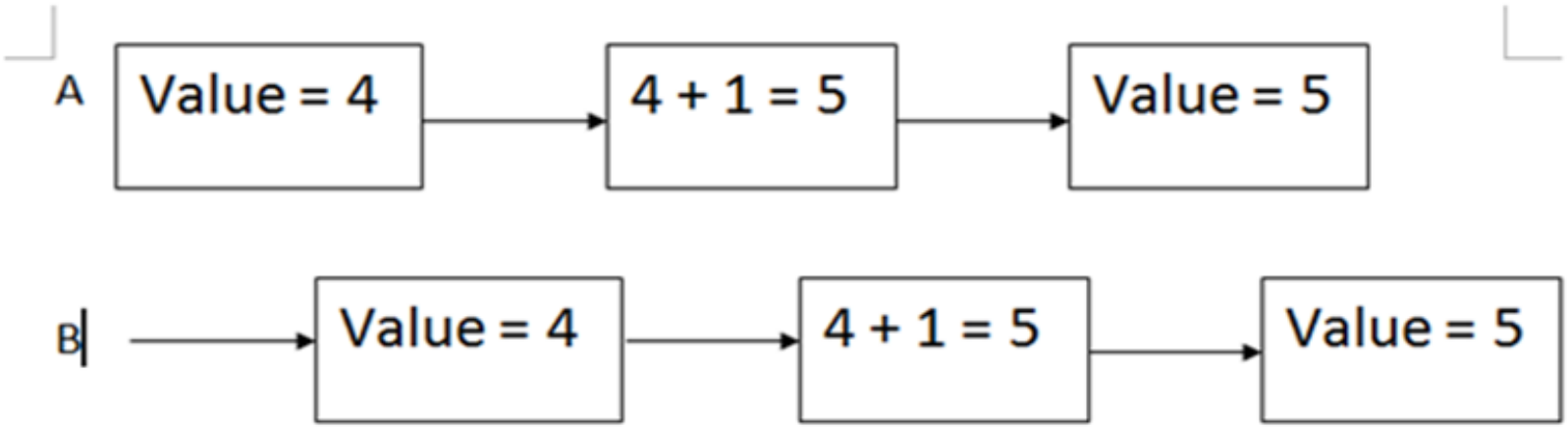
- ```
  int value = 0;
  public int getNext() {
    return ++value;
  }
  ```

- Works perfectly fine in a single-threaded application, but not in multi-threaded application.
  ```

Issue with Threads



Issue with Threads



- This can be fixed by adding “synchronized” keyword in the method heading.

Parallel Programming in Java

- Developers need to keep track of nitty-gritty details when working directly with threads.

Parallel Programming in Java

- Developers need to keep track of nitty-gritty details when working directly with threads.
- Language designers want programmers to write parallel programs without mastering these low level details.

Parallel Programming in Java

- Developers need to keep track of nitty-gritty details when working directly with threads.
- Language designers want programmers to write parallel programs without mastering these low level details.
- New concurrency libraries were added with JDK 5, 6 and 7.

Fork/Join Framework

- The framework is designed to make divide-and-conquer algorithms easy to parallelize.

Fork/Join Framework

- The framework is designed to make divide-and-conquer algorithms easy to parallelize.

```
Result solve(Problem problem) {
    if (problem is small)
        directly solve problem
    else {
        split problem into independent parts
        fork new subtasks to solve each part
        join all subtasks
        compose result from subresults
    }
}
```

Fork/Join Framework

- The class should either extends RecursiveAction (result-less) or RecursiveTask (result-bearing) class to enable it to use ForkJoin framework. (Other options discussed in report)

Fork/Join Framework

- The class should either extends RecursiveAction (result-less) or RecursiveTask (result-bearing) class to enable it to use ForkJoin framework. (Other options discussed in report)

- Use fork() method to divide the task and join() method to wait for the other thread to finish their task.

Fork/Join Example

```
class Fibonacci extends RecursiveTask<Integer> {
  final int n;
  Fibonacci(int n) { this.n = n; }
  Integer compute() {
    if (n <= 1)
       return n;
    Fibonacci f1 = new Fibonacci(n - 1);
    f1.fork();
    Fibonacci f2 = new Fibonacci(n - 2);
    return f2.compute() + f1.join();
  }
}
```

Fork/Join Concept

- Thread A finished with its work queue. What to do now ??

Fork/Join Concept

- Thread A finished with its work queue. What to do now ??
- There still exists some work that needs to be done.

Fork/Join Concept

- Thread A finished with its work queue. What to do now ??
- There still exists some work that needs to be done.
- Thread A takes the task exist in other threads work queue - Work Stealing.

Parallel Array

- For common operations such as Sorting, Searching, Aggregation etc. JDK 7 provides an easier way - ParallelArray classes.

Parallel Array

- For common operations such as Sorting, Searching, Aggregation etc. JDK 7 provides an easier way - ParallelArray classes.
- This provides an in-built method which does the task in parallel and returns the result.

Parallel Array Example

```
ParallelArray<Student> students = new ParallelArray<Student>(fjPool, data);
double bestGpa = students.withFilter(isSenior)
                         .withMapping(selectGpa)
                         .max();

public class Student {
    String name;
    int graduationYear;
    double gpa;
}

static final Ops.Predicate<Student> isSenior = new Ops.Predicate<Student>() {
    public boolean op(Student s) {
        return s.graduationYear == Student.THIS_YEAR;
    }
};

static final Ops.ObjectToDouble<Student> selectGpa = new Ops.ObjectToDouble<Student>() {
    public double op(Student student) {
        return student.gpa;
    }
};
```

Kiwiplan Framework

- Let's look at how the Kiwiplan framework is being structured.

Kiwiplan Framework

- Let's look at how the Kiwiplan framework is being structured.
- One of the major pattern used in the framework is the MVC pattern.

Kiwiplan Framework

- Let's look at how the Kiwiplan framework is being structured.
- One of the major pattern used in the framework is the MVC pattern.
- This pattern separates the modeling of the domain, the presentation, and the event handling.

Kiwiplan Framework

- The request made by the user goes through the pipeline in the bottom up fashion.

Kiwiplan Framework

- The request made by the user goes through the pipeline in the bottom up fashion.
- The request flows up the chain, and when each method returns the data flows back down.

Kiwiplan Framework

- The request made by the user goes through the pipeline in the bottom up fashion.
- The request flows up the chain, and when each method returns the data flows back down.
- Due to MVC pattern, the actual algorithms work with the pointer to the data but not with the actual data.

RowVisibility Filter

- Sometimes the users are interested in viewing only those records which satisfy certain criteria.

RowVisibility Filter

- Sometimes the users are interested in viewing only those records which satisfy certain criteria.
- Mapping from i^{th} row index of the interface to the j^{th} row in the database.

Name	Age	City
Bob	25	Auckland
Alice	20	Wellington
Samuel	37	Napier
John	27	Auckland
Kathy	29	Auckland

Mapping
1 ---> 1
2 ---> 2
3 ---> 3
4 ---> 4
5 ---> 5

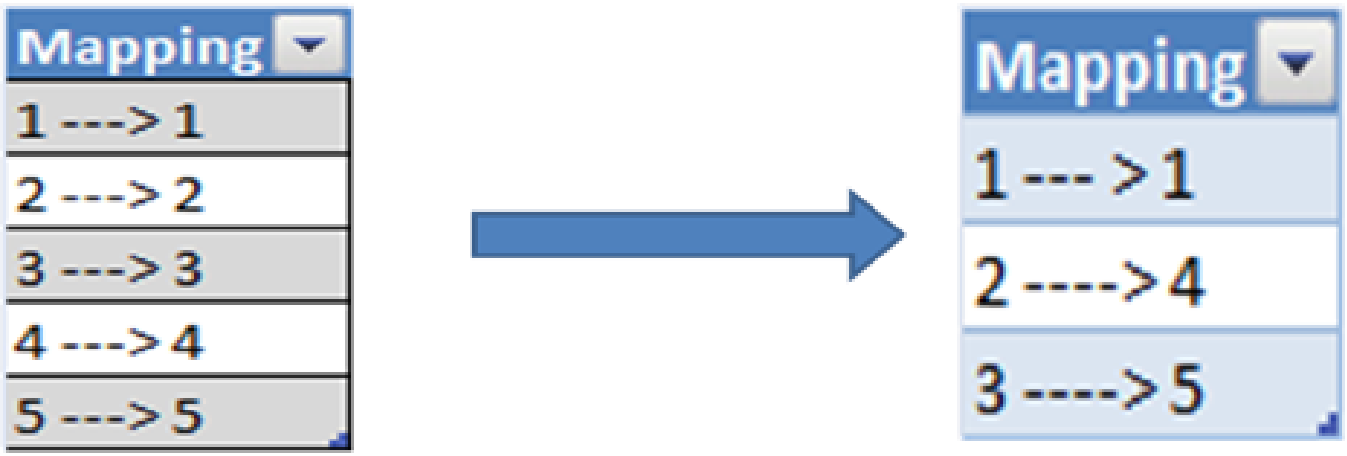
RowVisibility Filter

- We only want to view those rows where the age of the person is between 25 and 30 (incl.)

Name	Age	City
Bob	25	Auckland
Alice	20	Wellington
Samuel	37	Napier
John	27	Auckland
Kathy	29	Auckland

Name	Age	City
Bob	25	Auckland
John	27	Auckland
Kathy	29	Auckland

RowVisibility Filter



RowVisibility Sequential Algorithm

```
private void sequentialRecursiveTaskNoMerge(int low, int high, RowMapper mapper) {
    if (high - low <= STOP_DIVIDING) {
        for (int i = low; i <= high; i++) {
            if (shouldDisplayRow(i)) {
                mapper.setRowMapping(mapper.size(), i);
            }
        }
        return;
    }

    int middle = (high + low) / 2;
    sequentialRecursiveTaskNoMerge(low, middle, mapper);
    sequentialRecursiveTaskNoMerge(middle+1, high, mapper);
}
```

RowVisibility Parallel Algorithm

```
class FilterTaskNoMerge extends RecursiveAction {
    int low;
    int high;
    RowMapper mapper;

    public FilterTaskNoMerge(int low, int high, RowMapper mapper) {
        this.low = low;
        this.high = high;
        this.mapper = mapper;
    }

    @Override
    protected void compute() {
        if (high - low <= SEQUENTIAL_THRESHOLD) {
            for (int i = low; i <= high; i++) {
                if (shouldDisplayRow(i)) {
                    synchronized (this.mapper) {
                        mapper.setRowMapping(mapper.size(), i);
                    }
                }
            }
            return;
        }
        int middle = (high + low) / 2;
        invokeAll(new FilterTaskNoMerge(low, middle, mapper), new FilterTaskNoMerge(middle + 1, high, mapper));
    }
}
```

Row Visibility Analysis

- Time is measured in milliseconds and results are on dual core computer.

Problem Size	Sequential Running Time	Parallel Running Time
100	0	29
1000	2	28
10000	4	35
100000	22	76
1000000	450	112
5000000	1312	765

Sorting Filter

- One of the most common operation used by the users.

Last Name	First Name	Age	City
Paykel	Bob	28	Auckland
William	Kyle	23	Wellington
McCullum	Nathan	27	Auckland
Martin	Robin	22	Napier

1 ---> 1
2 ---> 2
3 ---> 3
4 ---> 4

Sorting Example

- Let's say we want to sort the table based on Last name. The table itself won't change.

Last Name	First Name	Age	City
Martin	Robin	22	Napier
McCullum	Nathan	27	Auckland
Paykel	Bob	28	Auckland
William	Kyle	23	Wellington

1 ---> 4
2 ---> 3
3 ---> 1
4 ---> 2

Sorting Algorithm

- Must be stable. Why?

Sorting Algorithm

- Must be stable. Why?

Name	Age
Bob	35
Bob	50

- Let's say the table has already been sorted according to "Age" column. Now, if we sort the table according to "Name" column, the order of the rows should remain the same.

Sorting Algorithm Sequentially

```
private void sequentialMergeSortWithTwoRowMapper(int low, int high, RowMapper from, RowMapper to) {
  if (high - low <= INSERTION_SORT_THRESHOLD) {
    insertionSortRowMapper(low, high, to);
    return;
  }

  int middle = (low + high) / 2;
  sequentialMergeSortWithTwoRowMapper(low, middle, to, from);
  sequentialMergeSortWithTwoRowMapper(middle + 1, high, to, from);

  if (array[from.mapRow(middle)].compareTo(array[from.mapRow(middle+1)]) <= 0) {
    for (int i = low; i <= high; i++) {
      to.setRowMapping(i, from.mapRow(i));
    }
    return;
  }

  //Merge step is similar to "MergeSort" Merge step
```

Sorting Algorithm Parallel

```
@Override
protected RowMapper compute() {
  if (high - low <= SEQUENTIAL_THRESHOLD) {
    RowMapper mapper = new RowMapper(high+1);
    sequentialMergeSortWithOneRowMapper(low, high, mapper);
    return mapper;
  }

  int middle = (low + high) / 2;
  ParallelSortTask leftTask = new ParallelSortTask(low, middle);
  leftTask.fork();
  ParallelSortTask rightTask = new ParallelSortTask(middle + 1, high);

  RowMapper right = rightTask.compute();
  RowMapper left = leftTask.join();
```

Sorting Filter Analysis

- Time is measured in milliseconds and results are on dual core computer.

Problem Size	Sequential Running Time	Parallel Running Time
100	2	31
1000	7	34
10000	25	56
100000	213	285
1000000	3500	2630
1500000	5326	3743

Future Work

- Still lots of work needs to be done.
- Refactor all the filter code which can possibly be run in parallel.
- In-dept knowledge of Java 7 once it will be released officially.

Questions

- QUESTIONS