Java Vector API

Marek Cuchý
marek.cuchy@fel.cvut.cz

B4M36ESW

April 25, 2022

Array Sum - Scalar Example

```
public static int sum(int[] array) {
    int sum = 0;
    for (int i : array) {
        sum += i;
    }
    return sum;
}
```

Array Sum – Vector Example

```
private static final VectorSpecies<Integer> SPECIES = IntVector.SPECIES_128;  10 usages
```

```
public static int sumVector(int[] array){
    int sum = 0;
    for (int offset = 0; offset < SPECIES.loopBound(array.length); offset += SPECIES.length()) {
        IntVector v = IntVector.fromArray(SPECIES, array, offset);
        sum += v.reduceLanes(VectorOperators.ADD);
    }
    return sum;
}
```

Array Sum – Vector Example Wrong?

```
private static final VectorSpecies<Integer> SPECIES = IntVector.SPECIES_128;  10 usages
```

```
public static int sumVector(int[] array){
    int sum = 0;
    for (int offset = 0; offset < SPECIES.loopBound(array.length); offset += SPECIES.length()) {
        IntVector v = IntVector.fromArray(SPECIES, array, offset);
        sum += v.reduceLanes(VectorOperators.ADD);
    }
    return sum;
}
```

What if the array length is not a multiple of the vector length (species)?

Array Sum – Vector Example Wrong?

```
private static final VectorSpecies<Integer> SPECIES = IntVector.SPECIES_128;  10 usages
```

```
public static int sumVector(int[] array){
    int sum = 0;
    for (int offset = 0; offset < SPECIES.loopBound(array.length); offset += SPECIES.length()) {
        IntVector v = IntVector.fromArray(SPECIES, array, offset);
        sum += v.reduceLanes(VectorOperators.ADD);
    }
    return sum;
}
```

What if the array length is not a multiple of the vector length (species)?

It would ignore the remainder of the array.

Array Sum – Vector Example Fixed

```
private static final VectorSpecies<Integer> SPECIES = IntVector.SPECIES_128;  10 usa
```

```
public static int sumVector(int[] array){
    int sum = 0;
    int offset = 0;
    for (; offset < SPECIES.loopBound(array.length); offset += SPECIES.length()) {
        IntVector v = IntVector.fromArray(SPECIES, array, offset);
        sum += v.reduceLanes(VectorOperators.ADD);
    }

    for (; offset < array.length; offset++) {
        sum += array[offset];
    }
    return sum;
}
```

You need to calculate the rest in a scalar fassion or…

Array Sum – Vector Example Fixed v2

```
public static int sumVectorMasked(int[] array){
    int sum = 0;
    int offset = 0;
    for (; offset < SPECIES.loopBound(array.length); offset += SPECIES.length()) {
        IntVector v = IntVector.fromArray(SPECIES, array, offset);
        sum += v.reduceLanes(VectorOperators.ADD);
    }

    VectorMask<Integer> mask = SPECIES.indexInRange(offset, array.length);
    IntVector v = IntVector.fromArray(SPECIES, array, offset, mask);
    sum += v.reduceLanes(VectorOperators.ADD);
    return sum;
}
```

… or use a mask (from my experiments, it is slower)

Array Sum – Vector Example Improved

```
public static int vectorBetter(int[] array) {  1 usage
    IntVector sum = IntVector.zero(SPECIES);

    int offset = 0;
    for (; offset < SPECIES.loopBound(array.length); offset += SPECIES.length()) {
        IntVector v = IntVector.fromArray(SPECIES, array, offset);
        sum = sum.add(v);
    }

    int sumVal = sum.reduceLanes(VectorOperators.ADD);
    for (; offset < array.length; offset++) {
        sumVal += array[offset];
    }
    return sumVal;
}
```

Benchmark

Vector length – 128bit

Benchmark	(vectorSize)	Mode	Cnt	Score	Error	Units
ArraySumBenchmark.std	1000007	avgt	50	272,736	± 5,699	us/op
ArraySumBenchmark.vectorImproved	1000007	avgt	50	91,537	± 2,146	us/op
ArraySumBenchmark.vectorImprovedMasked	1000007	avgt	50	92,181	± 2,189	us/op
ArraySumBenchmark.vectorNaive	1000007	avgt	50	334,925	± 6,202	us/op

Vector length – 256bit

Benchmark	(vectorSize)	Mode	Cnt	Score	Error	Units
ArraySumBenchmark.std	1000007	avgt	50	284,300	± 17,288	us/op
ArraySumBenchmark.vectorImproved	1000007	avgt	50	69,256	± 2,793	us/op
ArraySumBenchmark.vectorImprovedMasked	1000007	avgt	50	69,688	± 3,003	us/op
ArraySumBenchmark.vectorNaive	1000007	avgt	50	206,461	± 2,035	us/op

What about 512bit vector registers?

Benchmark – 512bit

Vector length – 512bit

Benchmark	(vectorSize)	Mode	Cnt	Score	Error	Units
ArraySumBenchmark.std	1000007	avgt	50	288,955	± 11,648	us/op
ArraySumBenchmark.vectorImproved	1000007	avgt	50	1672,734	± 17,939	us/op
ArraySumBenchmark.vectorImprovedMasked	1000007	avgt	50	1646,036	± 20,355	us/op
ArraySumBenchmark.vectorNaive	1000007	avgt	50	1310,094	± 16,380	us/op

Benchmark – 512bit

Vector length – 512bit

Benchmark	(vectorSize)	Mode	Cnt	Score	Error	Units
ArraySumBenchmark.std	1000007	avgt	50	288,955	± 11,648	us/op
ArraySumBenchmark.vectorImproved	1000007	avgt	50	1672,734	± 17,939	us/op
ArraySumBenchmark.vectorImprovedMasked	1000007	avgt	50	1646,036	± 20,355	us/op
ArraySumBenchmark.vectorNaive	1000007	avgt	50	1310,094	± 16,380	us/op

CPU does not have 512bit registers
=> *graceful degradation:* at least the code is still working

Benchmark – Matrix multiplication

Right matrix always transformed to column based (transposed) beforehand

Benchmark	(size)	(vectorSize)	Mode	Cnt	Score	Error	Units
MatrixBenchmark.multiplyManual1D	607	128	avgt	150	24,020	± 0,446	ms/op
MatrixBenchmark.multiplyManual1D	607	256	avgt	150	17,349	± 0,232	ms/op
MatrixBenchmark.multiplyManual2D	607	128	avgt	150	22,857	± 0,355	ms/op
MatrixBenchmark.multiplyManual2D	607	256	avgt	150	16,701	± 0,185	ms/op
MatrixBenchmark.multiplyMask1D	607	128	avgt	150	26,049	± 0,177	ms/op
MatrixBenchmark.multiplyMask1D	607	256	avgt	150	19,569	± 0,223	ms/op
MatrixBenchmark.multiplyMask2D	607	128	avgt	150	26,882	± 0,384	ms/op
MatrixBenchmark.multiplyMask2D	607	256	avgt	150	24,360	± 0,270	ms/op
MatrixBenchmarkStd.multiply2D	607	N/A	avgt	150	33,952	± 0,210	ms/op
MatrixBenchmarkStd.multiply1D	607	N/A	avgt	150	37,387	± 0,165	ms/op

References

- JEPs 338, 414, 417: https://openjdk.java.net/jeps/417