Feedback-Guided Switch Statement Optimization

Peng Zhao and José Nelson Amaral

Department of Computing Science
University of Alberta

Oct 6, 2004

1 – Switch = Case Selection + Case Action

```
switch (key)
{
    case 1:
        ... //action 1 break;
    case 2:
        ... //action 2 break;
    case 3:
        ... //action 3 break;
    default:
        ... //default action
}
... // next statement
```

2 – Outline

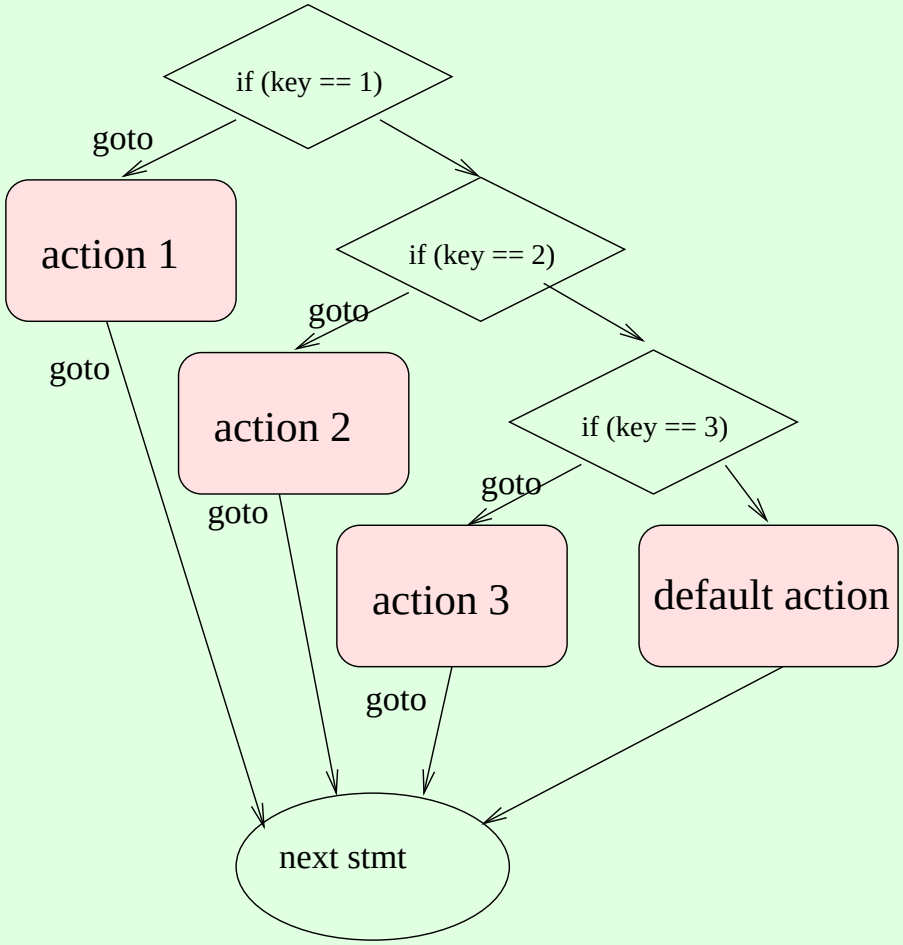
♠ Revisit the existent switch optimizations

♠ Propose two new techniques (hot default case promotion and large switch statement partition) to take advantage of skewed frequency among cases

♠ Investigate the potential of switch optimizations in Itanium-2 systems.

3 – Search Strategy (BR)

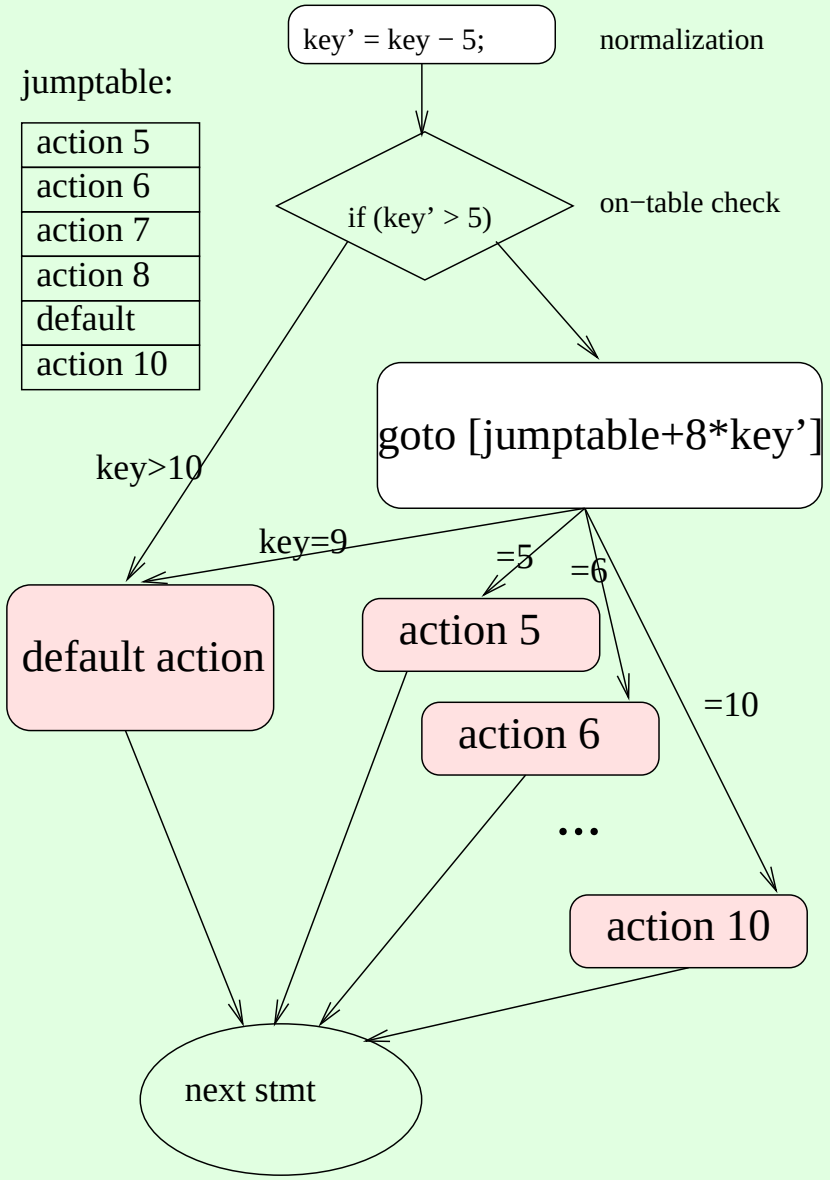
```
switch (key)
{
    case 1:
        ... //action 1 break;
    case 2:
        ... //action 2 break;
    case 3:
        ... //action 3 break;
    default:
        ... //default action
}
... // next statement
```



4 – Jump-Table Strategy (JT)

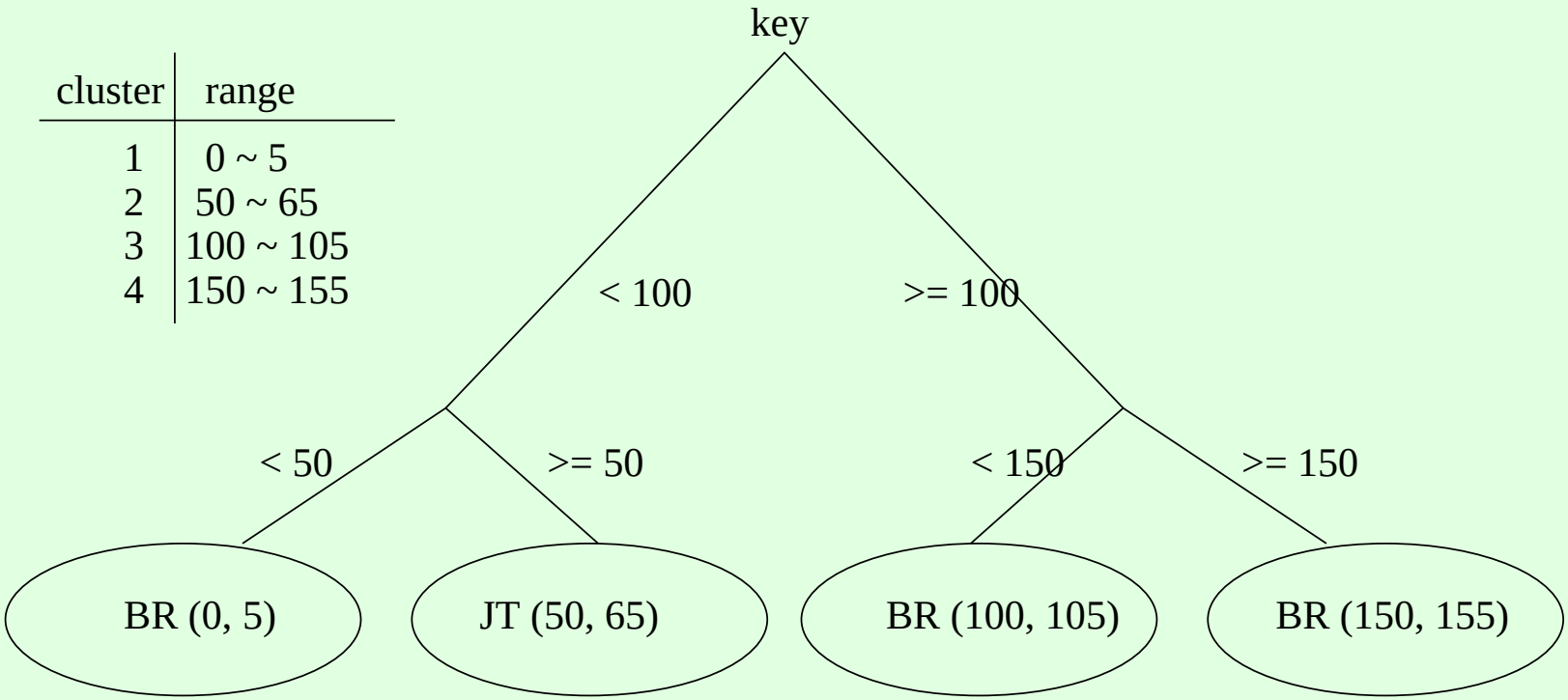
Feedback-guided Swtich Statement Optimization

```
switch (key)
{
    case 5:
        ... //action 5  break;
    case 6:
        ... //action 6  break;
    case 7:
        ... //action 7  break;
    case 8:
        ... //action 8  break;
    case 10:
        ... //action 10  break;
    default:
        ... //default action
}
```



5 – Hybrid Strategy (COMB)

cluster	range
1	0 ~ 5
2	50 ~ 65
3	100 ~ 105
4	150 ~ 155



6 – What if we have runtime feedback?

7 – Hot case hoisting (HH)

```
switch(key)
{
    case 5:
        ... break; // hot
    ... // Other cases
    case 9:
        ... break; // hot
    default:
        ... //default action
}
... // next statement
```

```
if(key == 5)
    ... // goto action5
if(key == 9)
    ... // goto action9
switch (key)
{
    ... // Other cases
    default:
        ... //default action
}
... // next statement
```

8 – What is missing?

♣ Selection for cases in the default category is still un-optimized

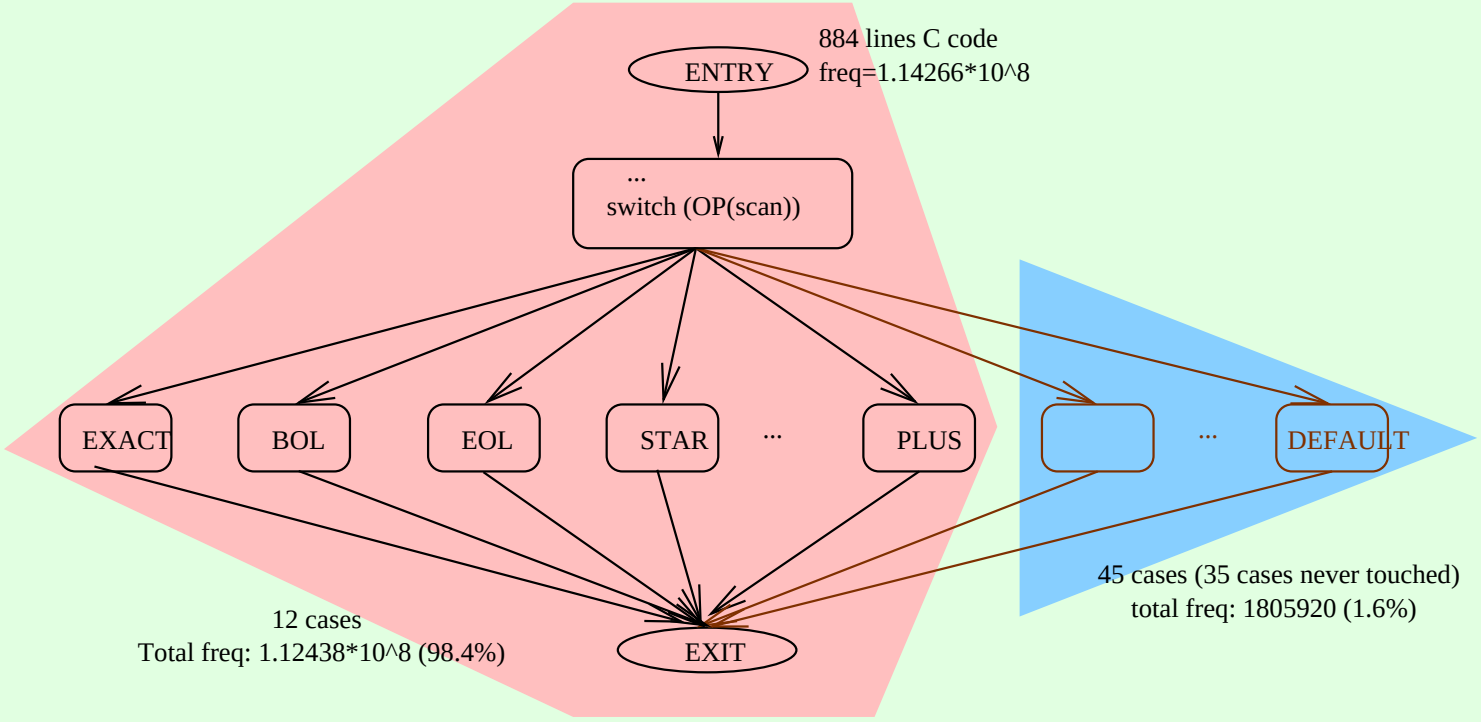
♣ The hot cases and cold cases are mixed together, which is bad for cache efficiency, and might prevent fast case selection and other optimizations such as inlining.

9 – Hot Default Case Promotion (DP)

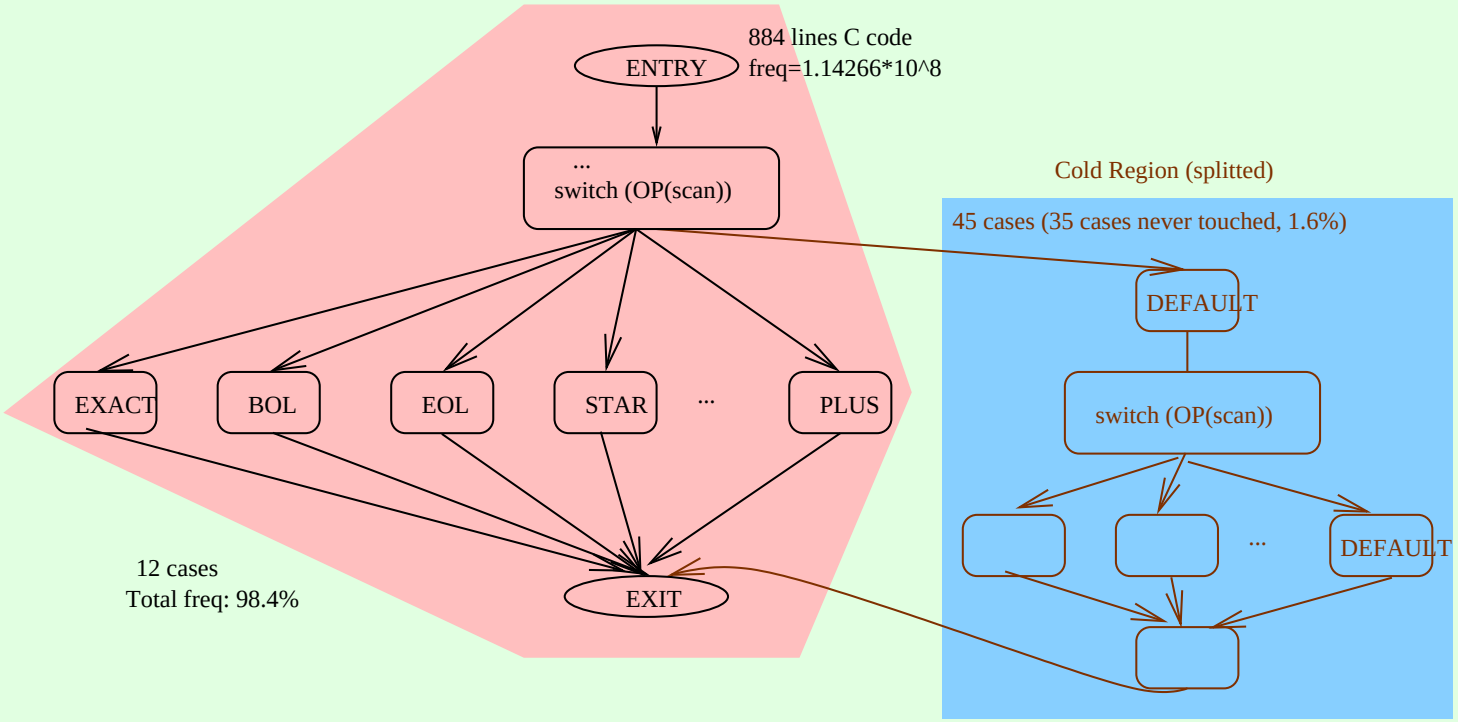
```
switch (key)
{
    case 5:
        ... break;
    case 6:
        ... break;
    case 7:
        ... break;
    default: // 9 is very hot
        ... //default action
}
... // next statement
```

```
switch (key)
{
    case 5:
        ... break;
    case 6:
        ... break;
    case 7:
        ... break;
    case 9: // case 9 is promoted
    default:
        ... //default action
}
... // next statement
```

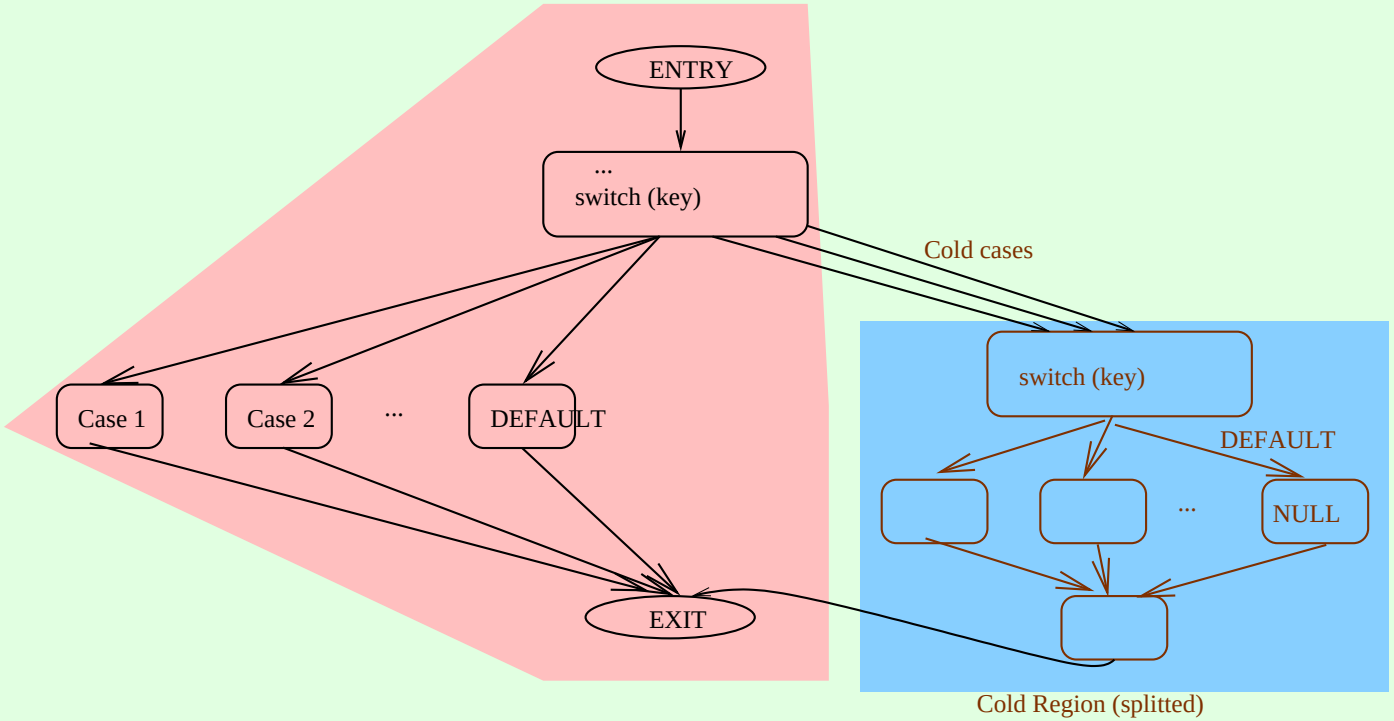
10 – Switch Partition(SP)



11 – Switch Partition(SP) (with cold default)



12 – Switch Partition(SP) (with hot default)



13 – Implementation

♠ Use the Open Research Compiler (ORC) 2.1 as the platform
♠ Extend profiling library to record the values and frequencies of the default cases
♠ Insert DP & SP before traditional switch optimization

$$\frac{\sum_{j=1}^{MaxPromotionNum} DefaultFreq[j]}{S.total\_freq} \geqslant PromotionThreshold \quad (1)$$

$$\frac{Size\_Ratio}{Freq\_Ratio} \geqslant Benefit\_Threshold \quad (2)$$

where:

$$\frac{ColdSize}{total\_size} = Size\_Ratio \qquad \frac{ColdFreq}{total\_freq} = Freq\_Ratio \quad (3)$$

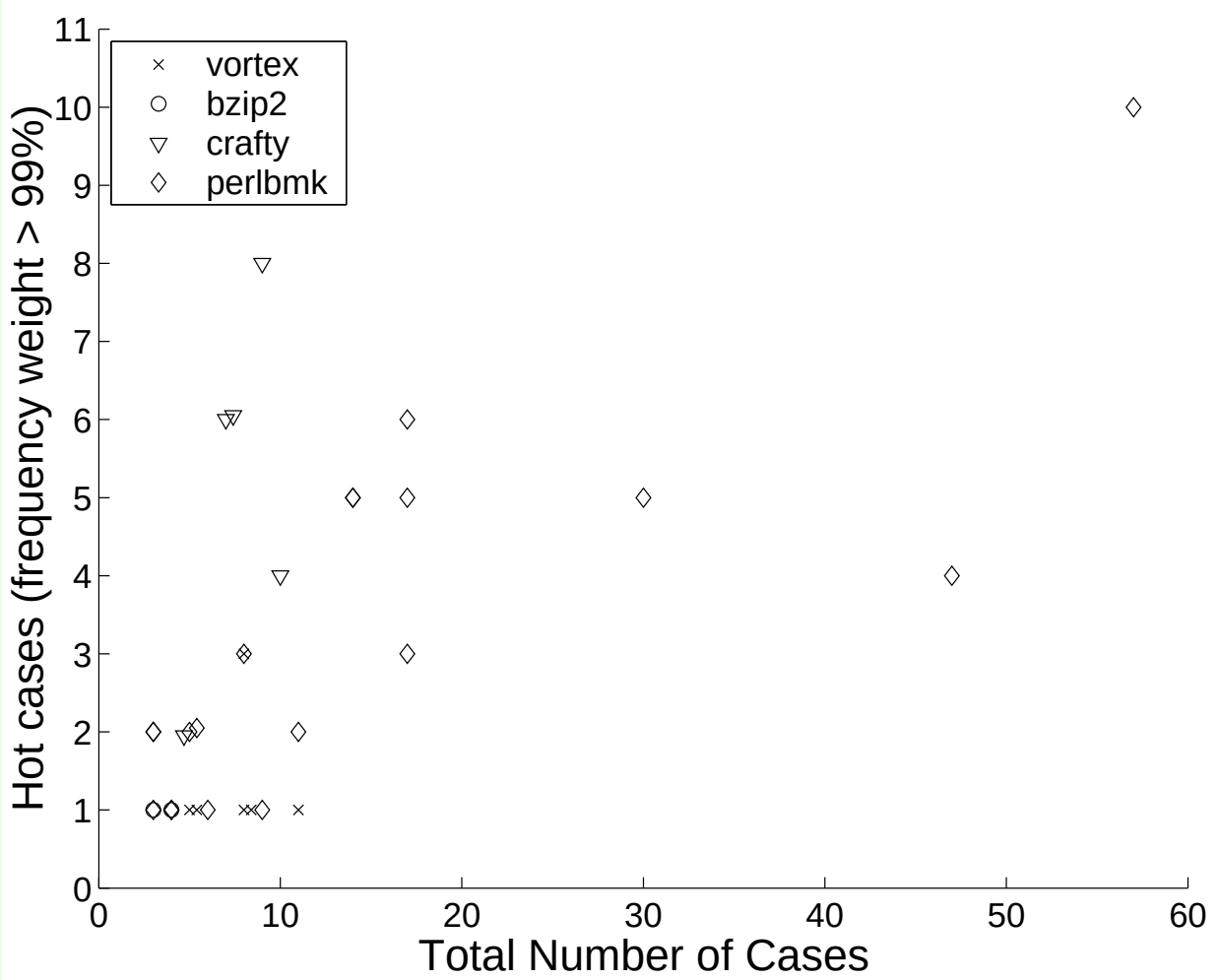
14 – Experimental Study

Feedback-guided Swtich Statement Optimization

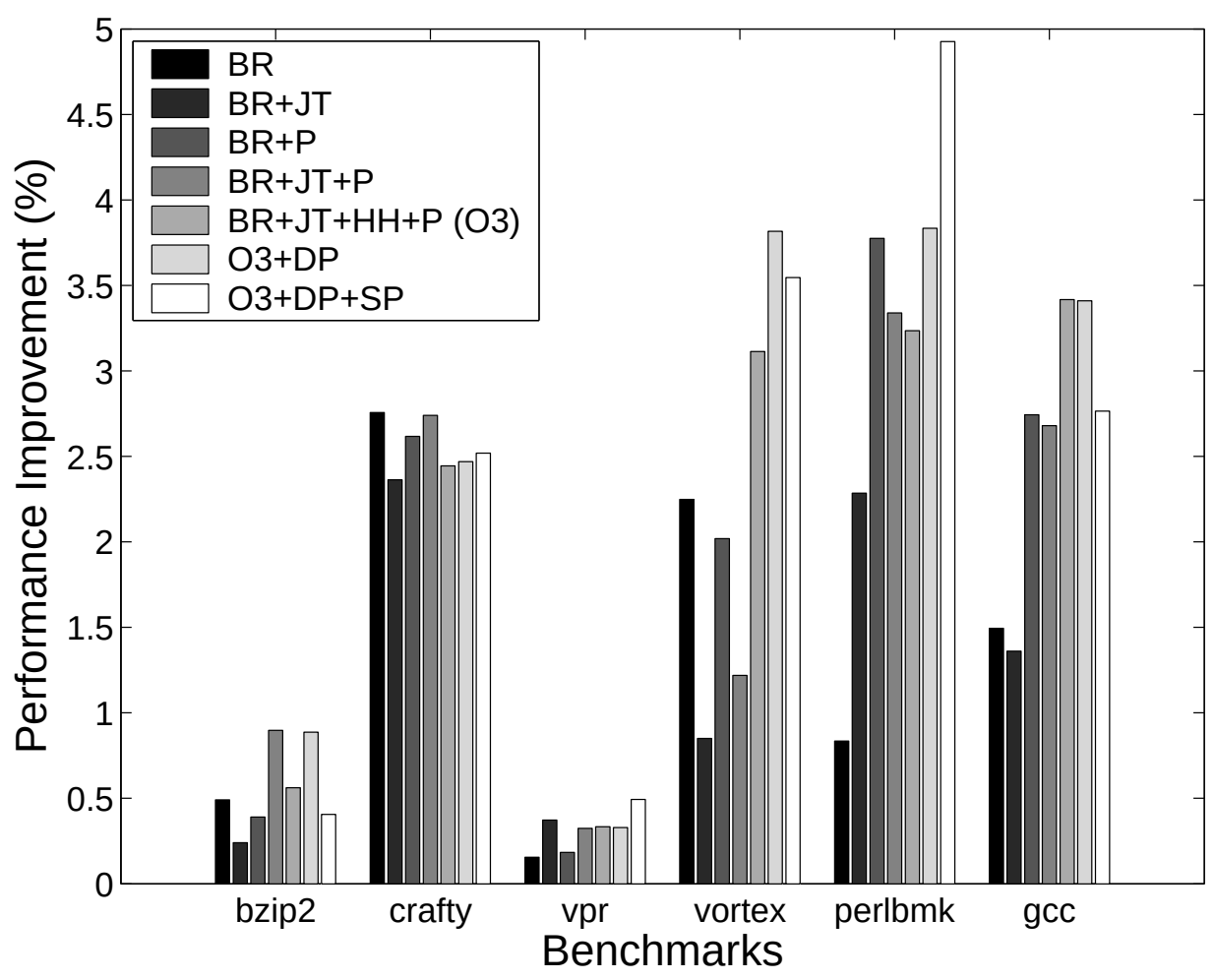
Benchmarks		bzip2	crafty	vpr	vortex	perl	gcc
# of Switches		3	42	12	37	127	374
Case Num Distr	<6	3	17	12	11	68	199
	$7 \sim 15$	0	25	0	22	30	117
	$16 \sim 30$	0	0	0	24	15	32
	$31 \sim 100$	0	0	0	0	11	21
	>100	0	0	0	0	3	5
Maximum Cases		4	13	6	30	243	398
Freq Distr	$10^6 \sim 10^7$	1	2	0	6	13	0
	$> 10^7$	1	3	0	0	7	0
Hot default		0	1	0	3	25	45

Table 1: Statistics of Switch-case Statements

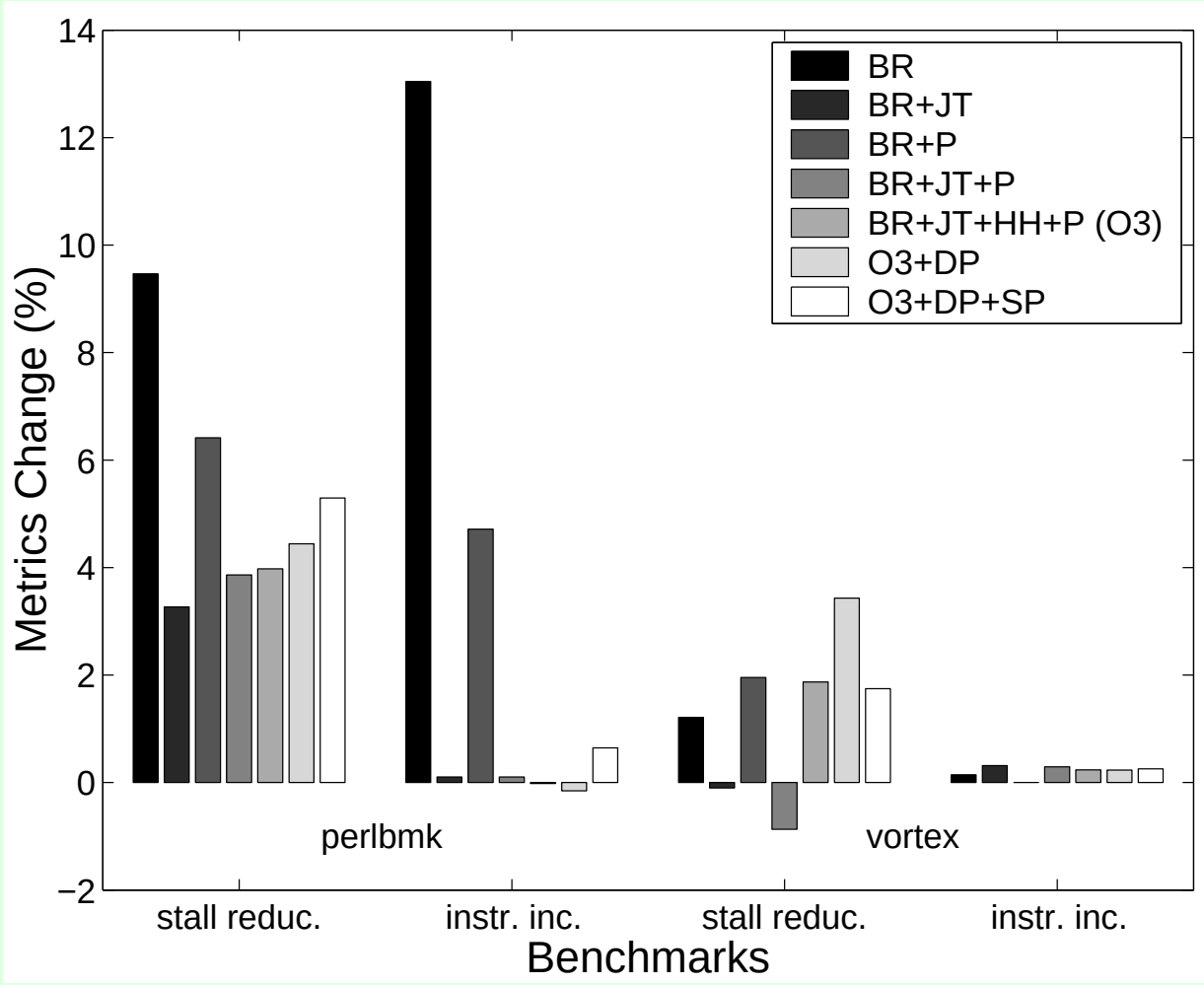
15 – Skewed frequency distribution among cases



16 – Runtime Performane Comparison



17 – Micro-architectural benchmarking



18 – Conclusion

♠ Switch optimization could yield non-trivial performance improvement if switch statements are invoked frequently in a program

♠ Feedback plays an important role in switch-case optimization

♠ Though BR strategy results in a lot more instructions than JT does, it often outperforms JT and JT+BR strategy

♠ Effectiveness of HH, DP and SP varies for different benchmarks.

Thank you very much!