The LoopPoint methodology in the gem5 Simulator

Zhantong Qiu

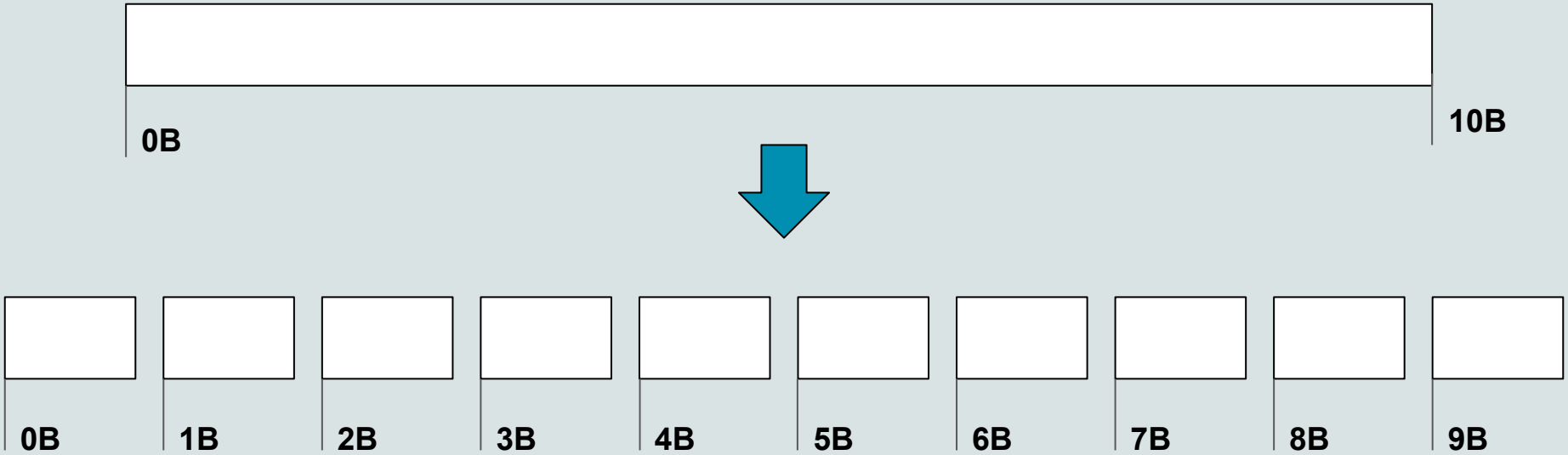
Outline

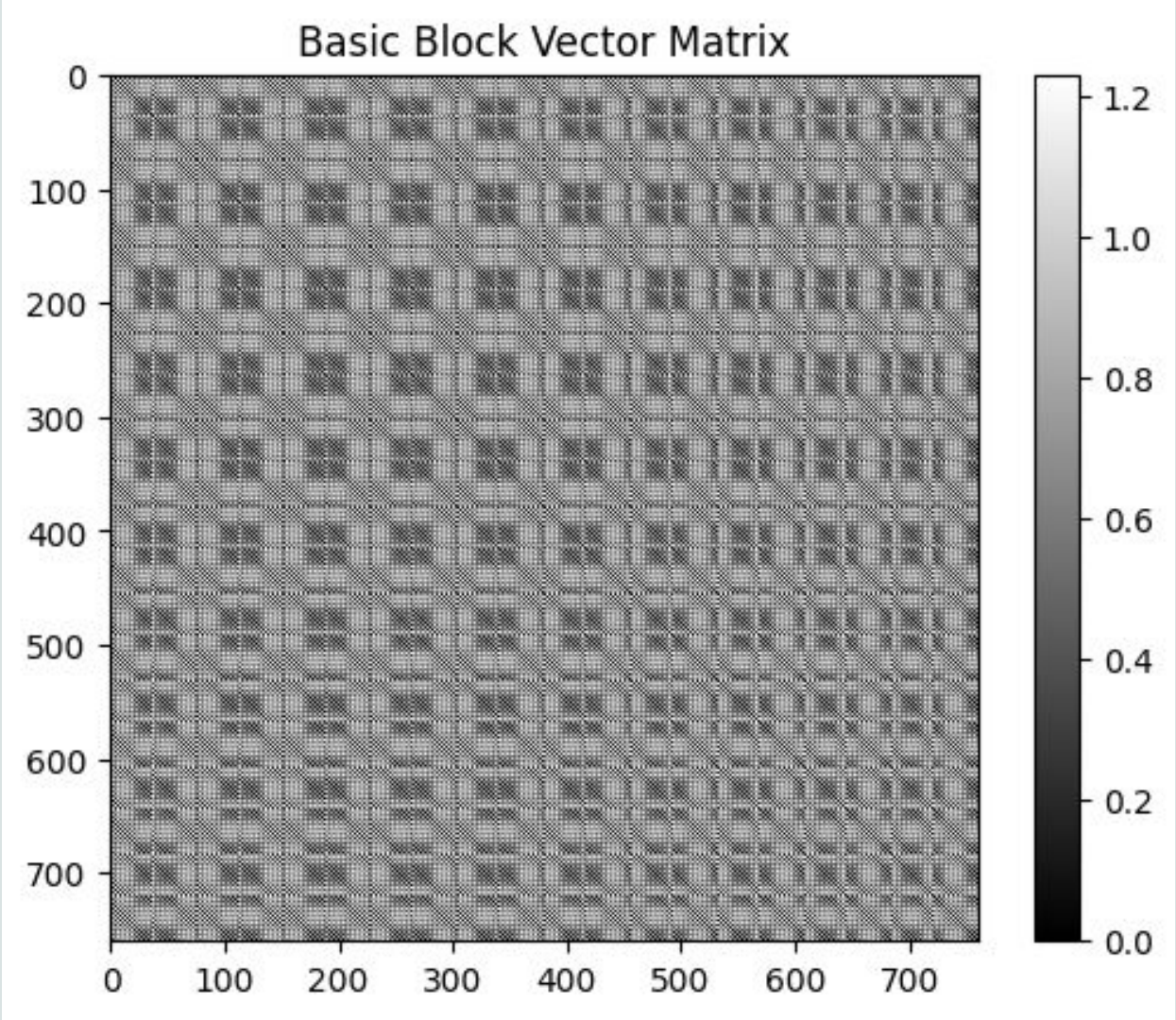
- **Introduction to SimPoint and LoopPoint Methodology**
- **LoopPoint methodology in the gem5 Simulator**
- **Other use cases of the LoopPoint methodology**
- **Summary and future work**

What is sampling?

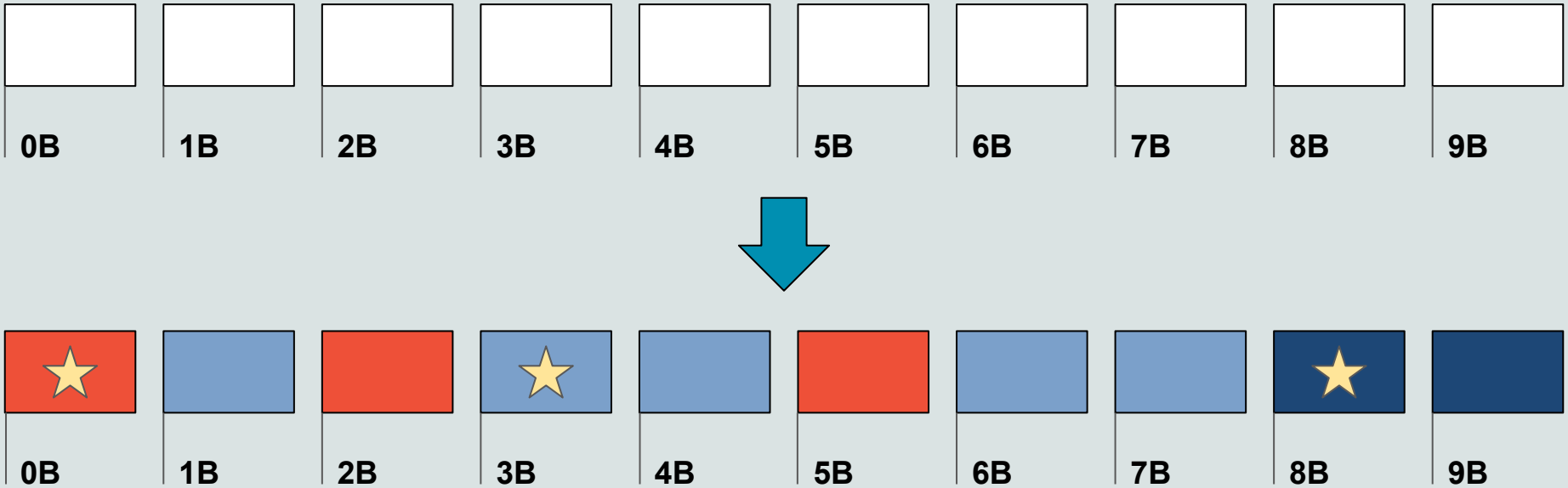
What is the SimPoint methodology?

The execution of the program is broken down into a number of intervals.



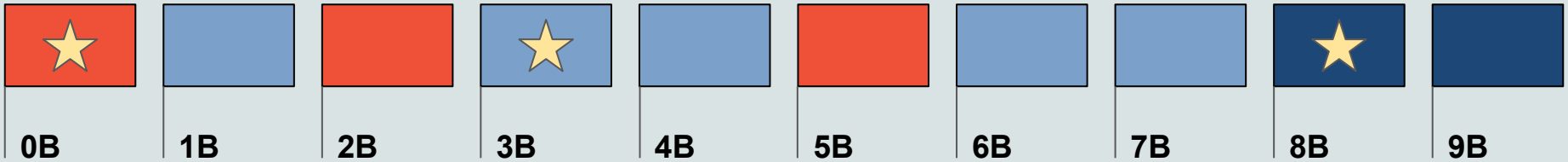

Basic Block Vector Matrix
0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700
1.2
1.0
0.8
0.6
0.4
0.2
0.0

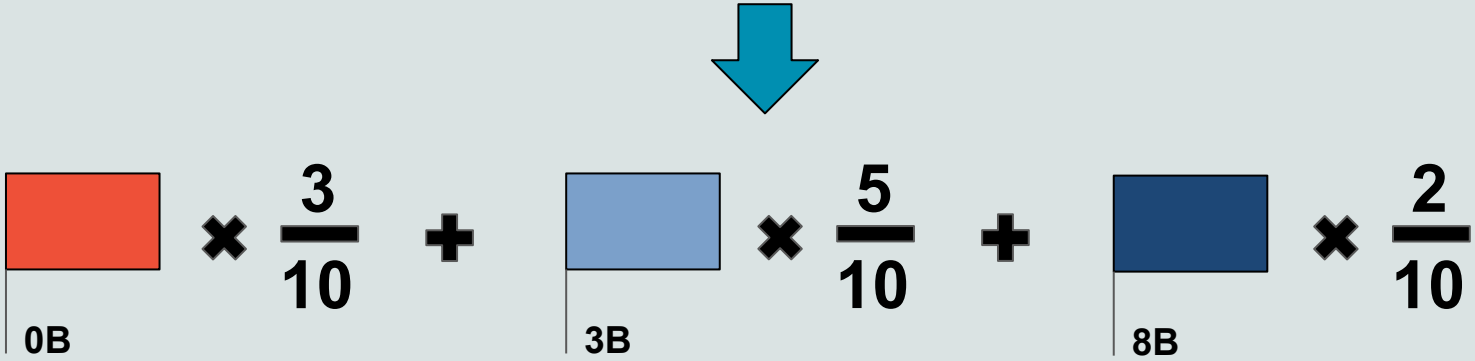
It uses the k-mean clustering algorithm to cluster the intervals into k clusters. Each cluster has one representative interval.



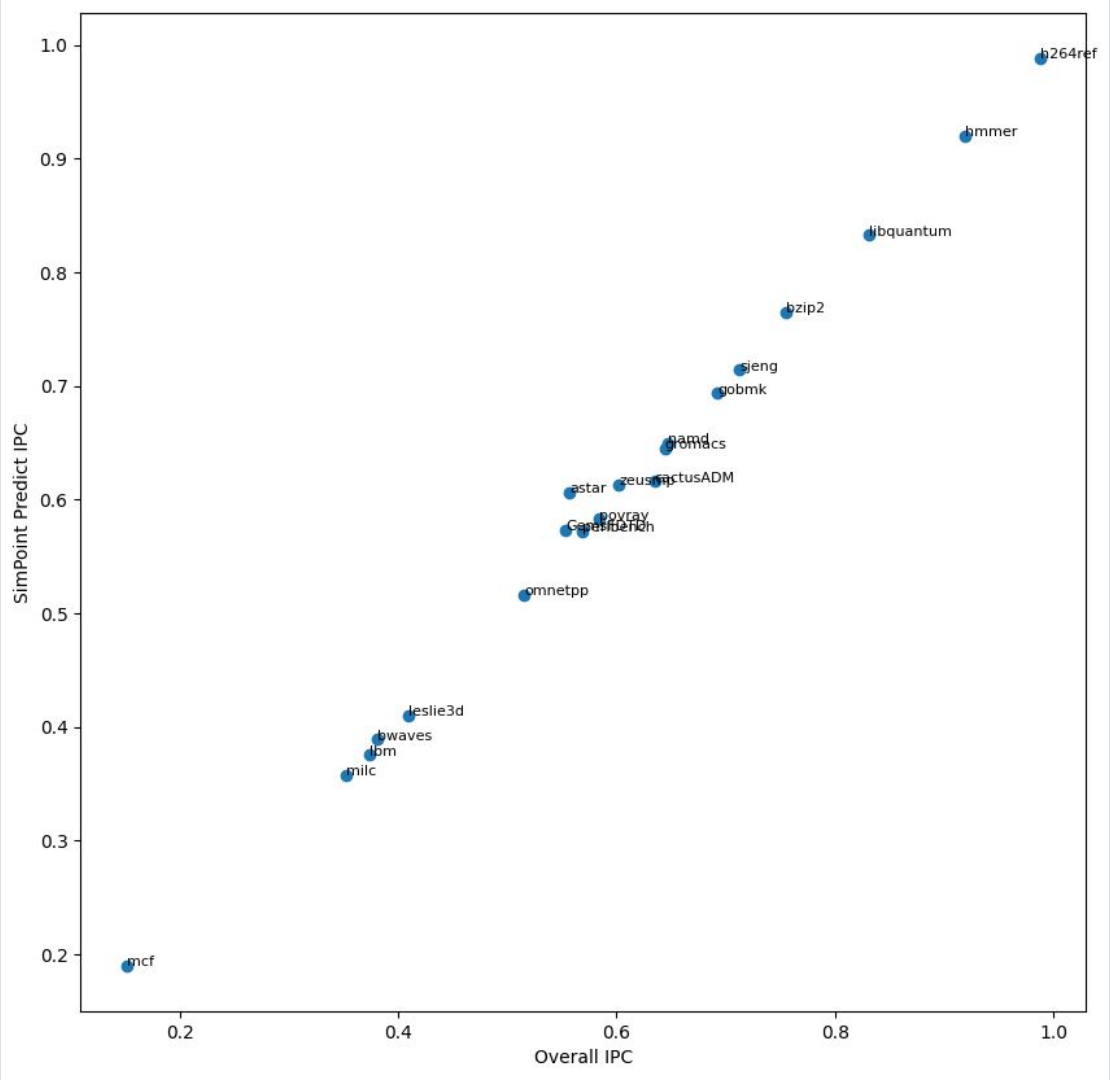
gem5

By performing detailed simulations on the representative interval of the clusters and applying weights to their IPC, we can predict IPC of simulating the whole program.

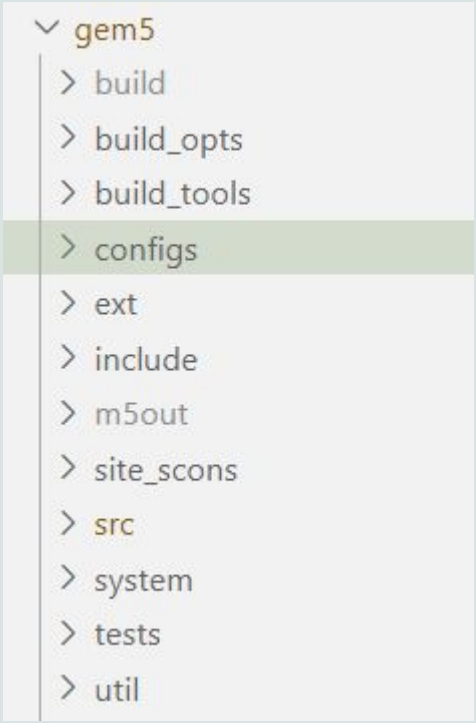


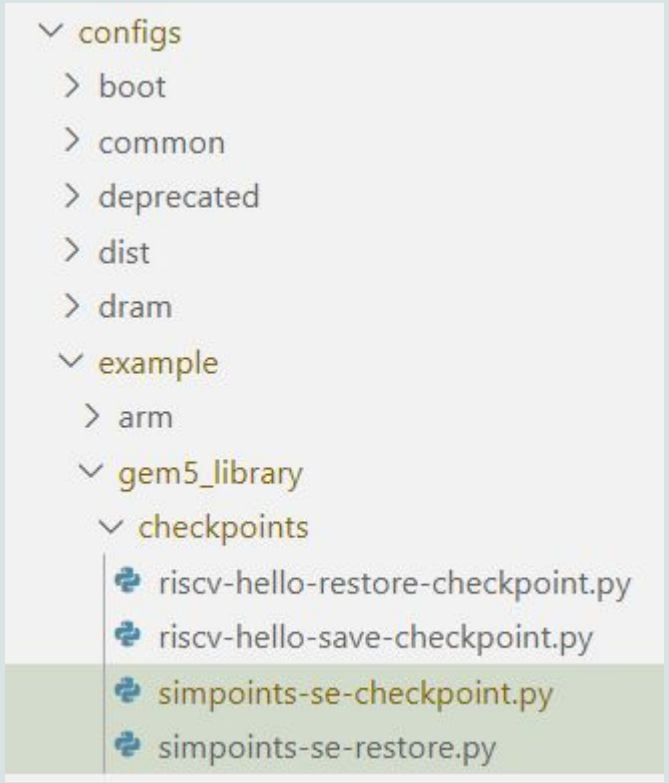


gem5


SimPoint Predict IPC
Overall IPC
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.4
0.6
0.8
1.0
h264ref
hmmer
libquantum
bzip2
sjeng
gobmk
namd
gromacs
cactusADM
astar
povray
omnetpp
leslie3d
bwaves
lbm
milc
mcf

Examples of using the SimPoint methodology can be found under the gem5 directory:





Limitation of SimPoint

- Unit of Work

- Execution Point Marker

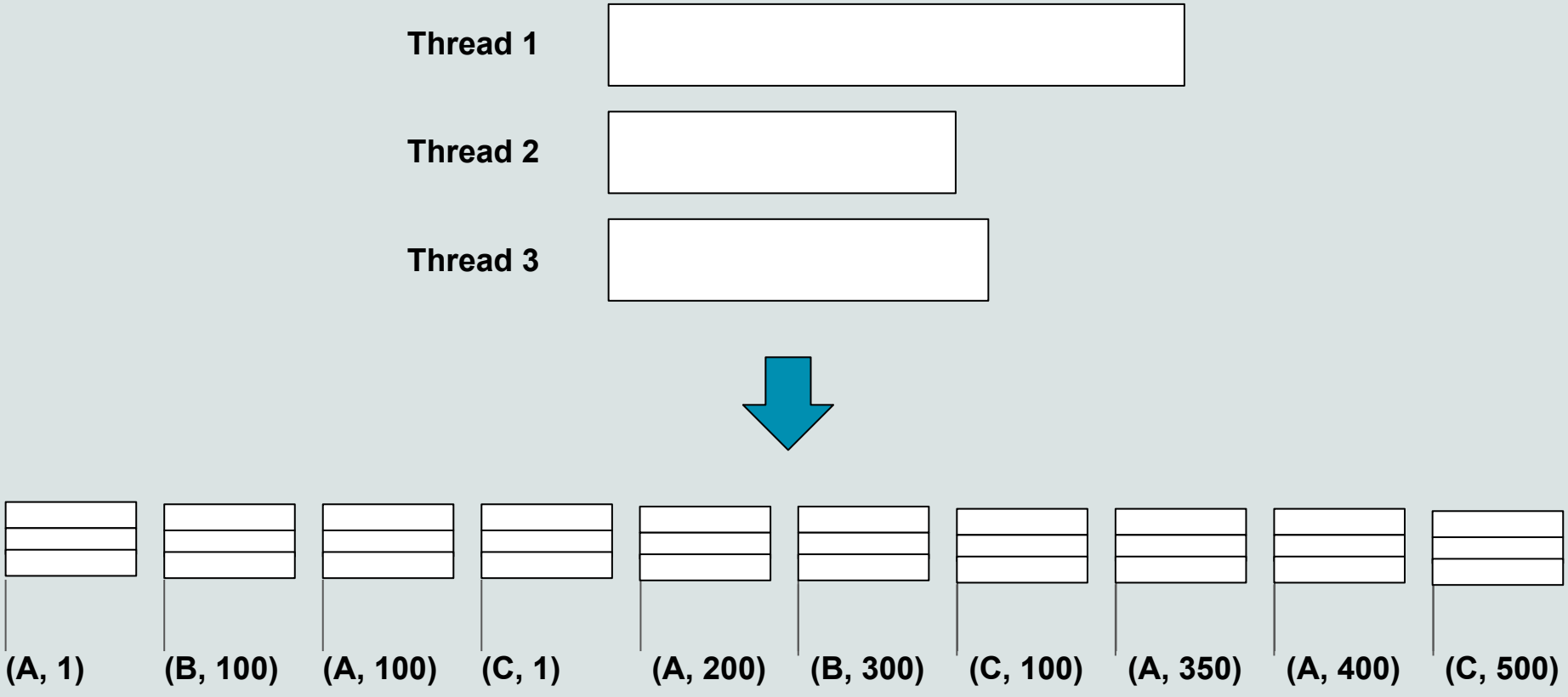
Limitation of SimPoint

- Unit of Work
- Execution Point Marker

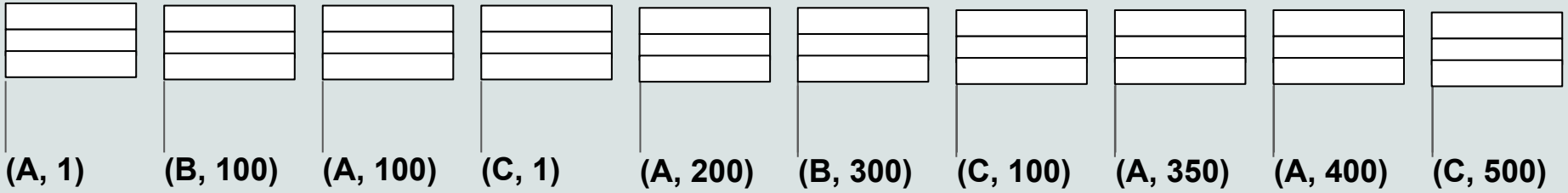
LoopPoint

- Using the number of loops iteration as the unit of work
- Using the PC Count pair as the execution point marker

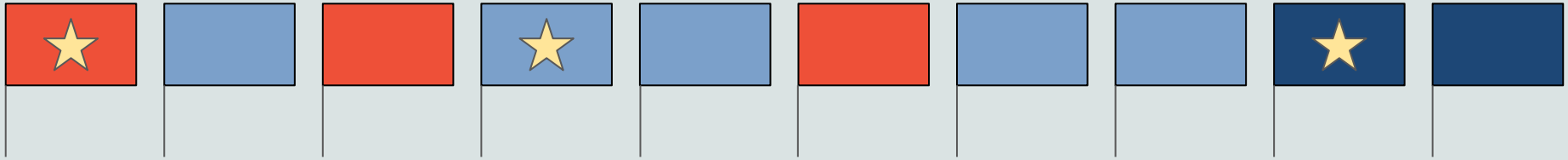
The LoopPoint methodology:



The LoopPoint methodology:



Region	PC	Count
1	A	1
2	B	100
3	A	100
4	C	1
5	A	200
…	…	…

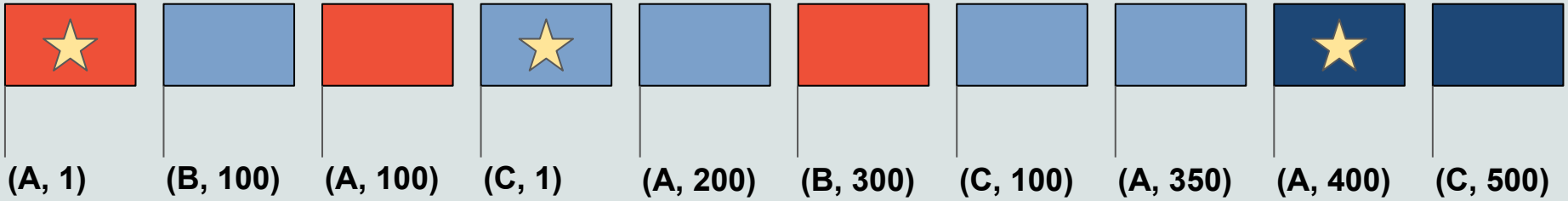


$$total\ runtime = \sum_{i=rep_1}^{rep_N} runtime_i \times multiplier_i$$

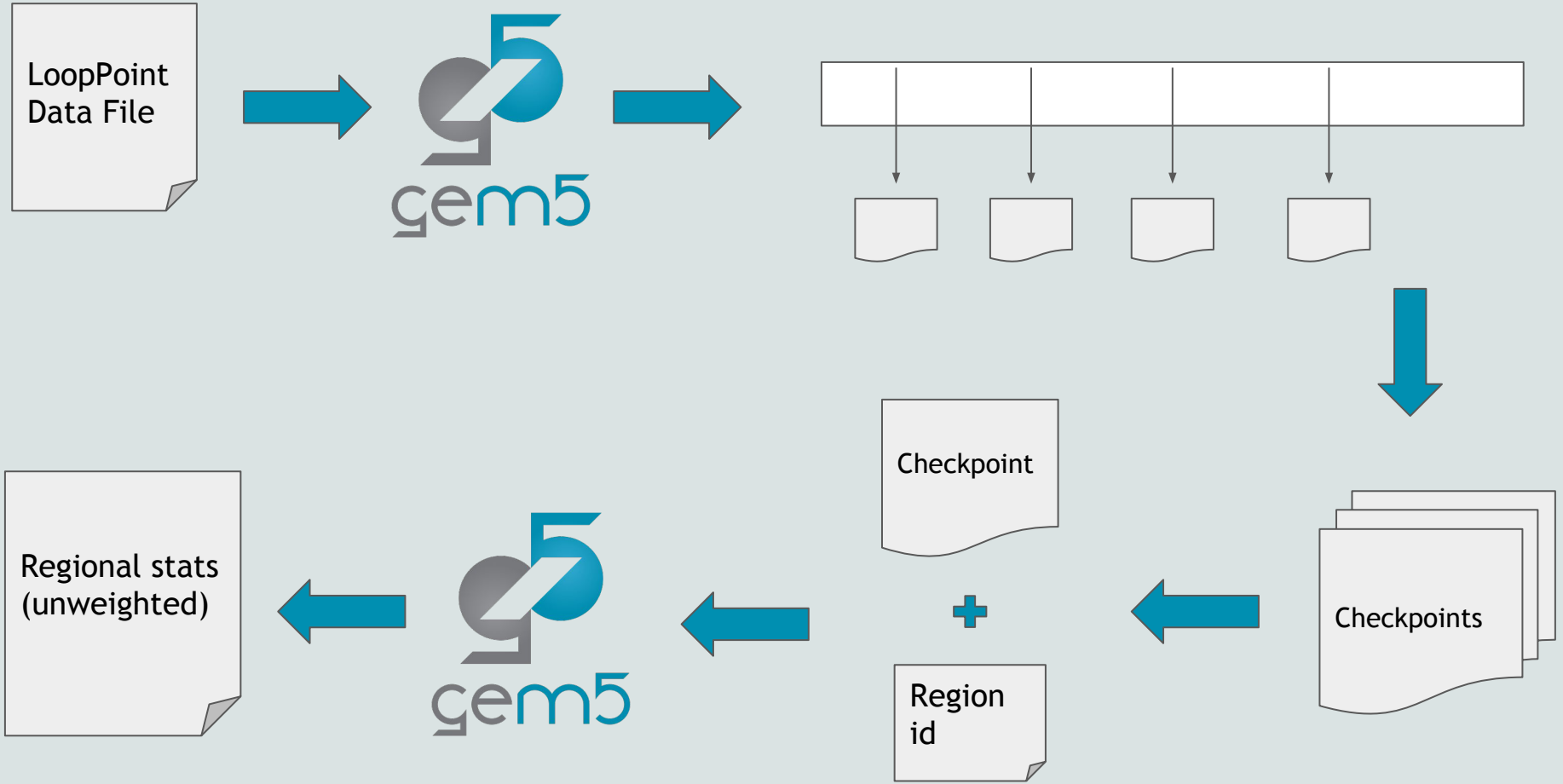
gem5

Source: Sabu et al., “LoopPoint: Checkpoint-driven Sampled Simulation for Multi-threaded Applications”, HPCA 2022

How to perform LoopPoint in gem5?

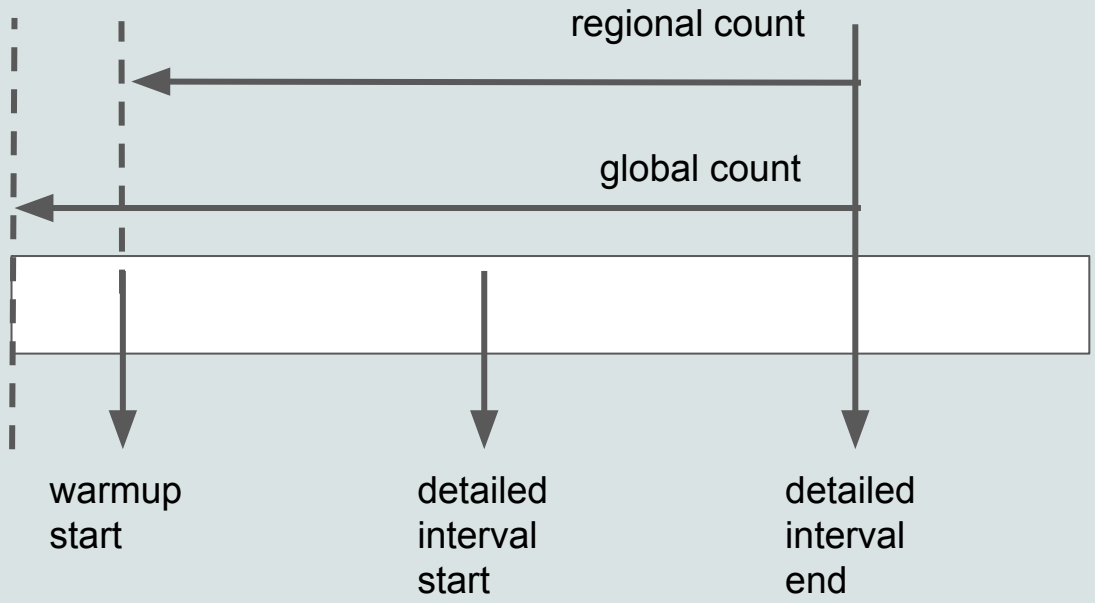


Region	PC	Count
1	A	1
2	B	100
3	A	100
4	C	1
5	A	200
…	…	…


LoopPoint
Data File
gem5
Checkpoint
Checkpoints
Region
id
gem5
Regional stats
(unweighted)

The LoopPoint JSON file

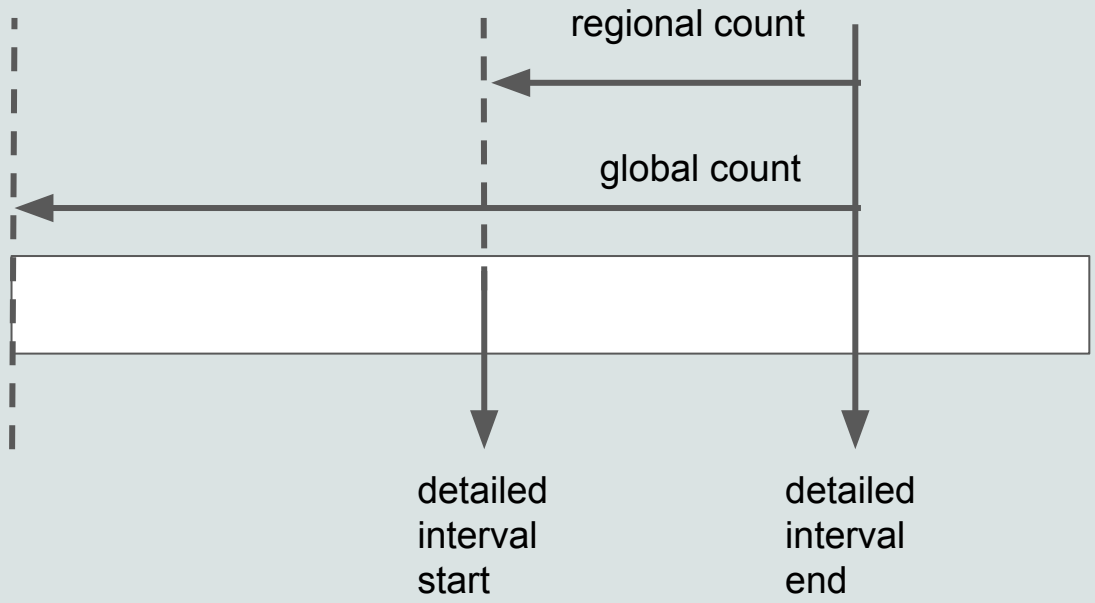
```
region id
    |___ detailed interval
    |       |___ start
    |       |       |___ pc
    |       |       |___ global
    |       |       |___ regional
    |       |___ end
    |               |___ pc
    |               |___ global
    |               |___ regional
    |___ warmup # optional to region
    |       |___ start
    |               |___ pc
    |               |___ global
    |
    |___ multiplier
```



gem5

The LoopPoint JSON file

```
region id
    |___ detailed interval
    |       |___ start
    |       |       |___ pc
    |       |       |___ global
    |       |       |___ regional
    |       |___ end
    |               |___ pc
    |               |___ global
    |               |___ regional
    |___ warmup # optional to region
    |       |___ start
    |               |___ pc
    |               |___ global
    |
    |___ multiplier
```



gem5

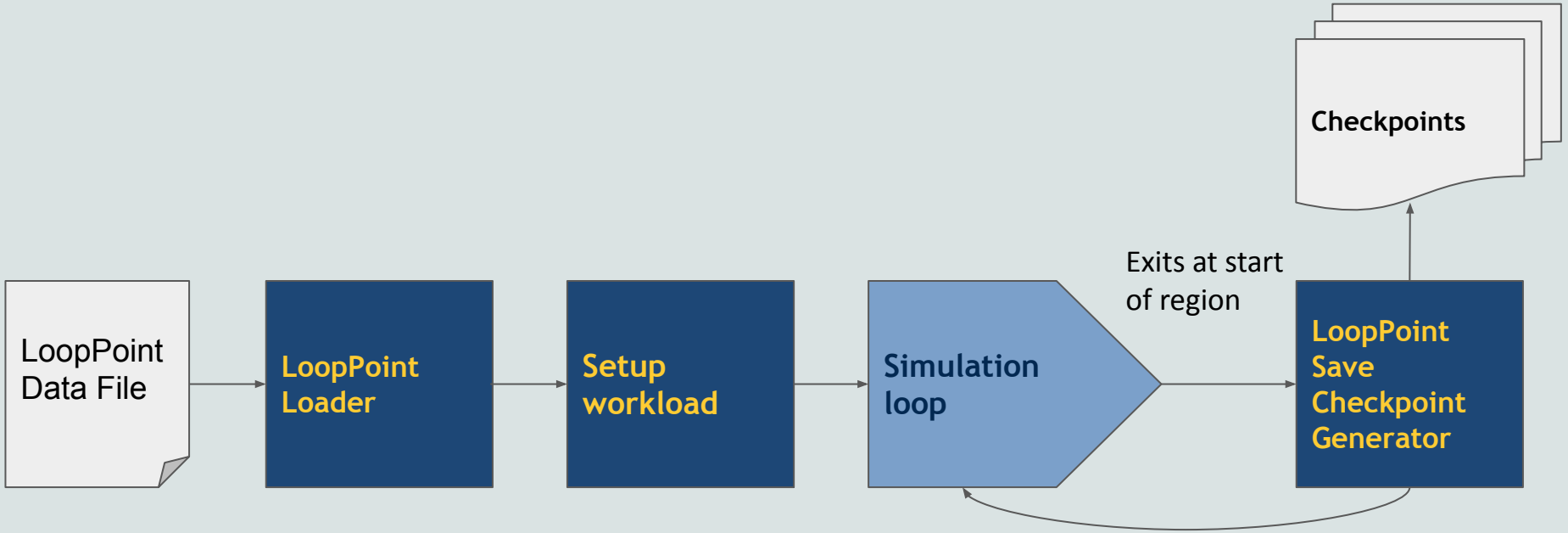
The LoopPoint JSON file

```
region id
    |___ detailed interval
    |       |___ start
    |       |       |___ pc
    |       |       |___ global
    |       |       |___ regional
    |       |___ end
    |               |___ pc
    |               |___ global
    |               |___ regional
    |___ warmup # optional to region
    |       |___ start
    |               |___ pc
    |               |___ global
    |
    |___ multiplier
```

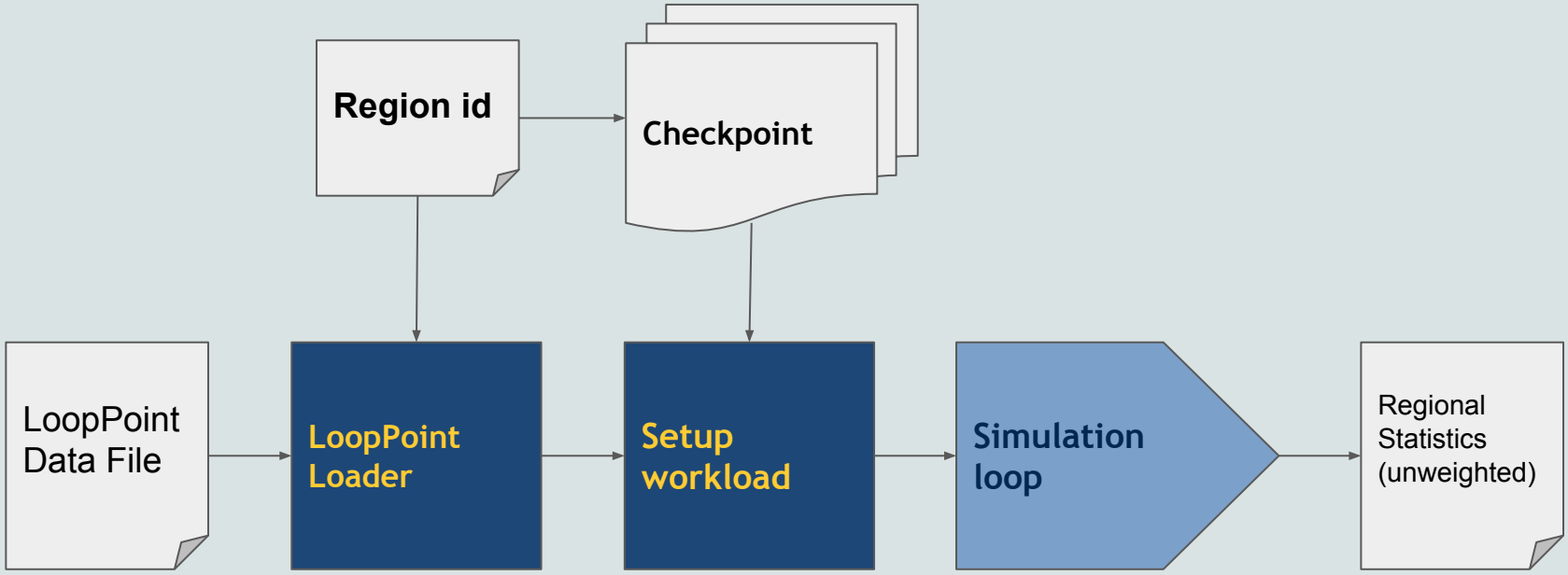
```
    "1": {
        "detailed interval": {
            "start": {
                "pc": #4221392,
                "global": 211076617,
                "regional": 15326617
            },
            "end": {
                "pc": #4221392,
                "global": 219060252,
                "regional": 23310252
            }
        },
        "multiplier": 4.0,
        "warmup": {
            "start": {
                "pc": #4221056,
                "count": 23520614
            }
        }
    },
```

```
    "2": {
            "detailed interval": {
                "start": {
                    "pc": #4206672,
                    "global": 1
                },
                "end": {
                    "pc": #4221392,
                    "global": 6861604,
                    "regional": 6861604
                }
            },
            "multiplier": 1.0
        }
```

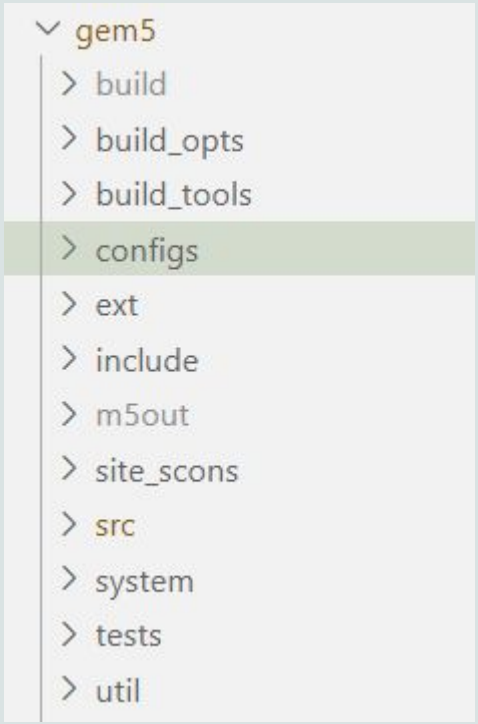
How to take checkpoints for LoopPoint sampling?

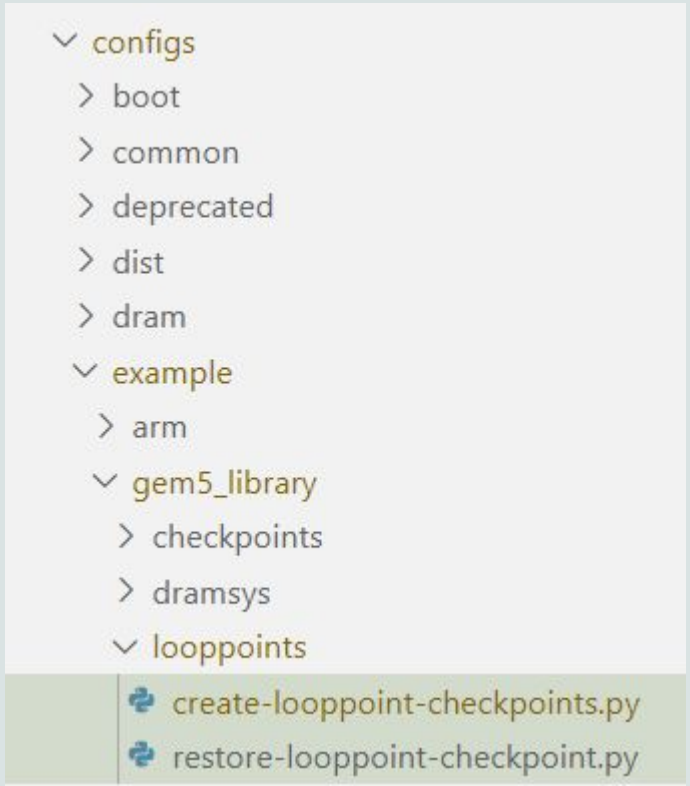


The process is similar for running the region



Examples using the LoopPoint methodology can be found under the gem5 directory:





Other use case

- The LoopPoint module uses the **PcCountTracker** to track the PC count pairs.
- **PcCountTracker** will raise an exit event at all the inputted PC Count pairs.
- For example, if we want to stop at **0x40086** when it has been executed **100** times:

```
from m5.params import PcCountPair
from m5.objects import PcCountTrackerManager
from gem5.components.processors.simple_processor import SimpleProcessor

processor = SimpleProcessor(
    cpu_type=CPUTypes.TIMING,
    isa=ISA.X86,
    num_cores=9,
)

target_pccountpairs = [PcCountPair(0x40086,100)]

manager = PcCountTrackerManager()
manager.targets = target_pccountpairs
for core in processor.get_cores():
    core.add_pc_tracker_probe(target_pccountpairs, manager)
```

Other use case

- The LoopPoint module uses the **PcCountTracker** to track the PC count pairs.
- **PcCountTracker** will raise an exit event at all the inputted PC Count pairs.
- For example, if we want to stop at **0x40086** when it has been executed **100** times:

```
from m5.params import PcCountPair
from m5.objects import PcCountTrackerManager
from gem5.components.processors.simple_processor import SimpleProcessor

processor = SimpleProcessor(
    cpu_type=CPUTypes.TIMING,
    isa=ISA.X86,
    num_cores=9,
)

target_pccountpairs = [PcCountPair(0x40086,100)]

manager = PcCountTrackerManager()
manager.targets = target_pccountpairs
for core in processor.get_cores():
    core.add_pc_tracker_probe(target_pccountpairs, manager)
```

Other use case

- The LoopPoint module uses the **PcCountTracker** to track the PC count pairs.
- **PcCountTracker** will raise an exit event at all the inputted PC Count pairs.
- For example, if we want to stop at **0x40086** when it has been executed **100** times:

```
from m5.params import PcCountPair
from m5.objects import PcCountTrackerManager
from gem5.components.processors.simple_processor import SimpleProcessor

processor = SimpleProcessor(
    cpu_type=CPUTypes.TIMING,
    isa=ISA.X86,
    num_cores=9,
)

target_pccountpairs = [PcCountPair(0x40086,100)]

manager = PcCountTrackerManager()
manager.targets = target_pccountpairs
for core in processor.get_cores():
    core.add_pc_tracker_probe(target_pccountpairs, manager)
```

Summary

Future works

- LoopPoint Analysis in the gem5 simulator is on the way
  - More information about its current status can be found:

    https://github.com/darchr/gem5/tree/looppointAnalysis

- Combine regions runs and improve statistic process