University of Central Florida

Department of Computer Science

CDA 5106: Spring 2025

Machine Problem 3: Dynamic Instruction Scheduling

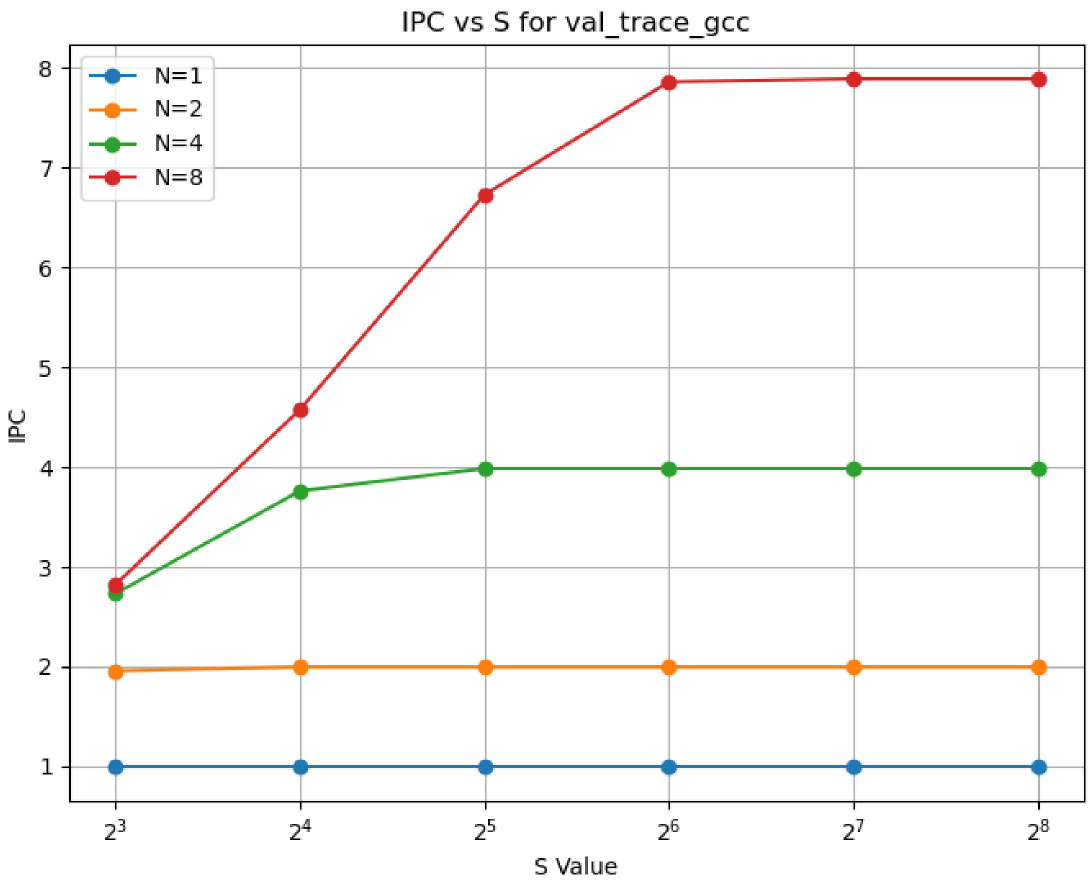
by

Elliott D’Amato, Gabriel Antonio Alvim D Arco, Athena Leong, Jacob Riesterer

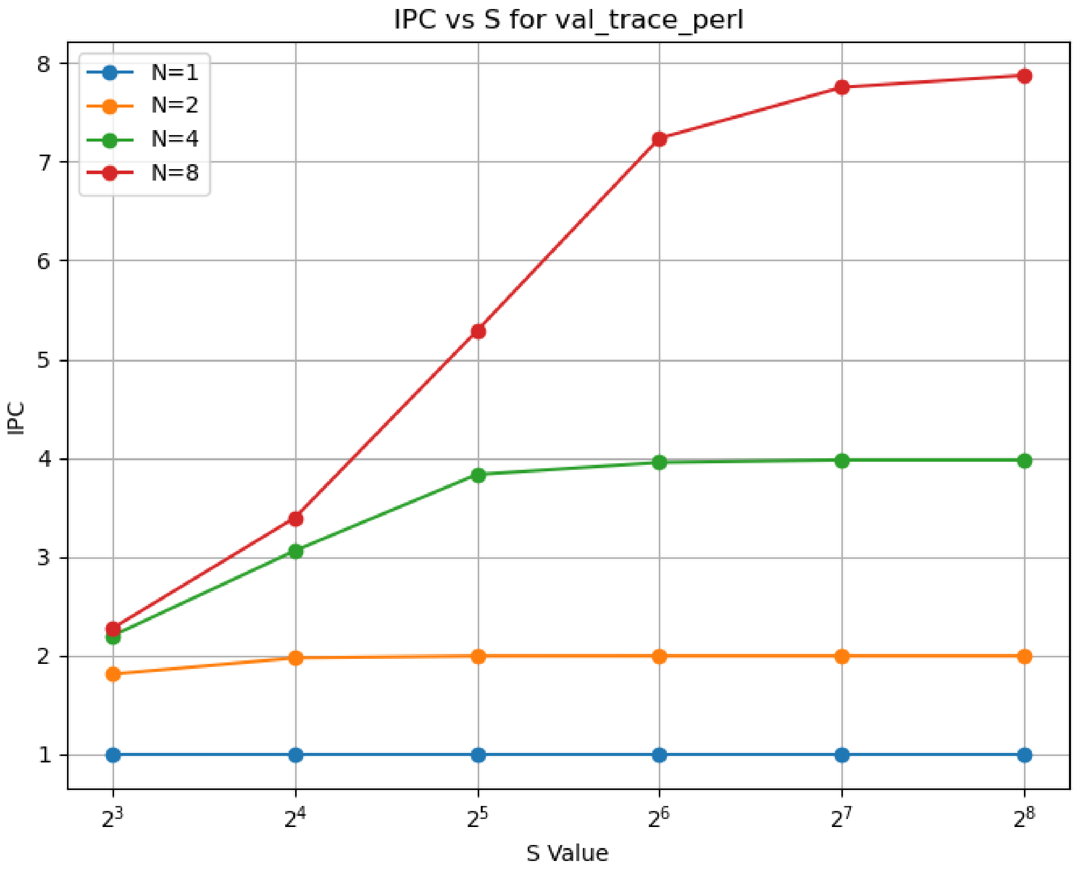
Honor Pledge: "I have neither given nor received unauthorized aid on this test or assignment."

Student’s electronic signature: _________Elliott D’Amato______
(sign by typing your name)
Student’s electronic signature: __Gabriel Antonio Alvim D Arco__
(sign by typing your name)
Student’s electronic signature: _________Athena Leong________
(sign by typing your name)
Student’s electronic signature: _________Jacob Riesterer_______
(sign by typing your name)

Question 1)



IPC vs S for val_trace_gcc
N=1
N=2
N=4
N=8
IPC
8
7
6
5
4
3
2
1
2^3
2^4
2^5
2^6
2^7
2^8
S Value


IPC vs S for val_trace_perl
N=1
N=2
N=4
N=8
IPC
S Value
1
2
3
4
5
6
7
8
2^3
2^4
2^5
2^6
2^7
2^8

Question 2)

Minimum S achieving 95% of max IPC for val_trace_gcc:
N=1: S=8
N=2: S=8
N=4: S=32
N=8: S=64

Minimum S achieving 95% of max IPC for val_trace_perl:
N=1: S=8
N=2: S=16
N=4: S=32
N=8: S=128

Optimized Scheduling Queue size per peak Fetch Rate		
	Benchmark = gcc	Benchmark = perl
N = 1	8	8
N = 2	8	16
N = 4	32	32
N = 8	64	128

Question 3)

A) **Given the goal of achieving an IPC close to the peak fetch rate, please explain the relationship between S and N.**

N, in our model, is the peak fetch rate. This is equal to the number of instructions that can be fetched and decoded in a cycle, which is the limiting value for our model's architecture. As such, our goal is to set a value S that allows for the IPC to approach N. As seen in both graphs, there is a direct relationship between S and the yielded IPC, where IPC increases as S increases. This relationship follows a sigmoidal curve with its upper horizontal asymptote being N. Increasing the scheduling queue size allows for more instructions to be potentially selected for execution, which then allows more instructions to be executed out-of-order and increases IPC. This does have diminishing returns, as the IPC can never exceed the value of N. In our two traces, it seems that an S value of 8 to 16 times the N value will yield a 95% high IPC value.

B) **Different benchmarks (traces) lead to different IPCs despite using the same architectures. What might be the reason?**

The major reason why different benchmarks lead to different IPCs is due to the hazards present in the traces. The traces may have the same number of operations, but their types and dependencies are not the same. As such, traces that perform worse on the superscalar processors are those that have many dependencies and hazards that restrict the ability to perform out-of-order execution. For example, a trace file that only runs one instruction of "addi $t1 $t1 1" would have a maximum IPC of 1 regardless of S or N as each instruction must be executed in order due to the WAW hazards.

Specifically for the two displayed benchmarks, it is likely that the gcc benchmark has either less hazards or less conflicting hazards than the perl benchmark. It is possible that there are more hazards present in the gcc benchmark, but they may be of smaller penalties. This would mean having more hazards of lower clock cycles (based on optype) which would yield an average penalty lower than infrequent but large hazards of a significant degree. It is also pertinent to note that the hazard placement does matter, as individual hazards are avoidable with a sufficient scheduling queue size, while grouped hazards make larger stalls across the superscalar architecture.