SE3910 – REAL TIME SYSTEMS

Performance Analysis

ROADMAP

- Wednesday
  - Performance Analysis
- Friday
  - Queuing Theory
- Monday
  - Memory Utilization

OBJECTIVES

- List the complexity of various real time related activities
- Explain the relationship between Amdahl's and Gustafson's laws
- Explain how gprof can be used to aid in analyzing program execution

Algorithms

↳ Sort methods

↳ Java tool

↳ visual VM

HOW DO WE DETERMINE THE PERFORMANCE OF A SYSTEM?

Big O ⇒ Measures algorithmic complexity

time ↪ versus ↩ design constraints

LD methods / tasks etc.

HOW DO WE DETERMINE THE PERFORMANCE OF A SYSTEM?

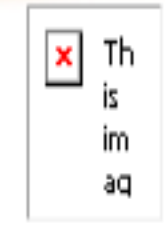
what kind of times do we care about?

⇒ Average

⇒ Max

⇒ Mins

⇒ Deviations



THEORETICAL CONCEPTS

Task	Complexity
When there are mutual exclusion constraints, finding a completely optimal run time scheduler	Impossible
Deciding whether it is possible to schedule a set of periodic tasks that only use semaphores to enforce mutual exclusion	NP-Hard
The multiprocessor scheduling problem with 2 processors, no resources, independent tasks, and arbitrary task completion times	NP-Complete
The multiprocessor scheduling problem with 2 processors, no resources, independent tasks, and arbitrary partial order, and task computation times of either 1 or 2 units of time.	NP-Complete
The multiprocessor scheduling algorithm with 2 processors, one resource, a forest partial order, and the computation time of every task equal to 1 unit	NP-Complete
The multiprocessor scheduling algorithm with 3 or more processors, one resource, all independent tasks, and each computation time equal to 1 unit	NP Complete
Earliest deadline first scheduling	No optimal in multiprocessing case

Related to scheduling

Very hard to do just at runtime.

- Who can explain Amdahl's law? ⇒ Computer Org

AMDAHL'S LAW

"Make the common case fast"

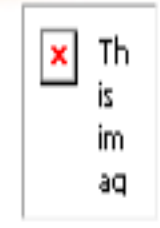
AMDAHL'S LAW

- For a constant problem size
- The incremental speedup approaches zero as the number of processor elements grows
  - S => Fraction of code which is of a serial nature
  - N => Number of processors available for serialization
- $Speedup_{Amdahl} = \frac{S+(1-S)}{S+\frac{(1-S)}{N}} = \frac{1}{\frac{1}{N}+\left(1-\frac{1}{N}\right)S}$

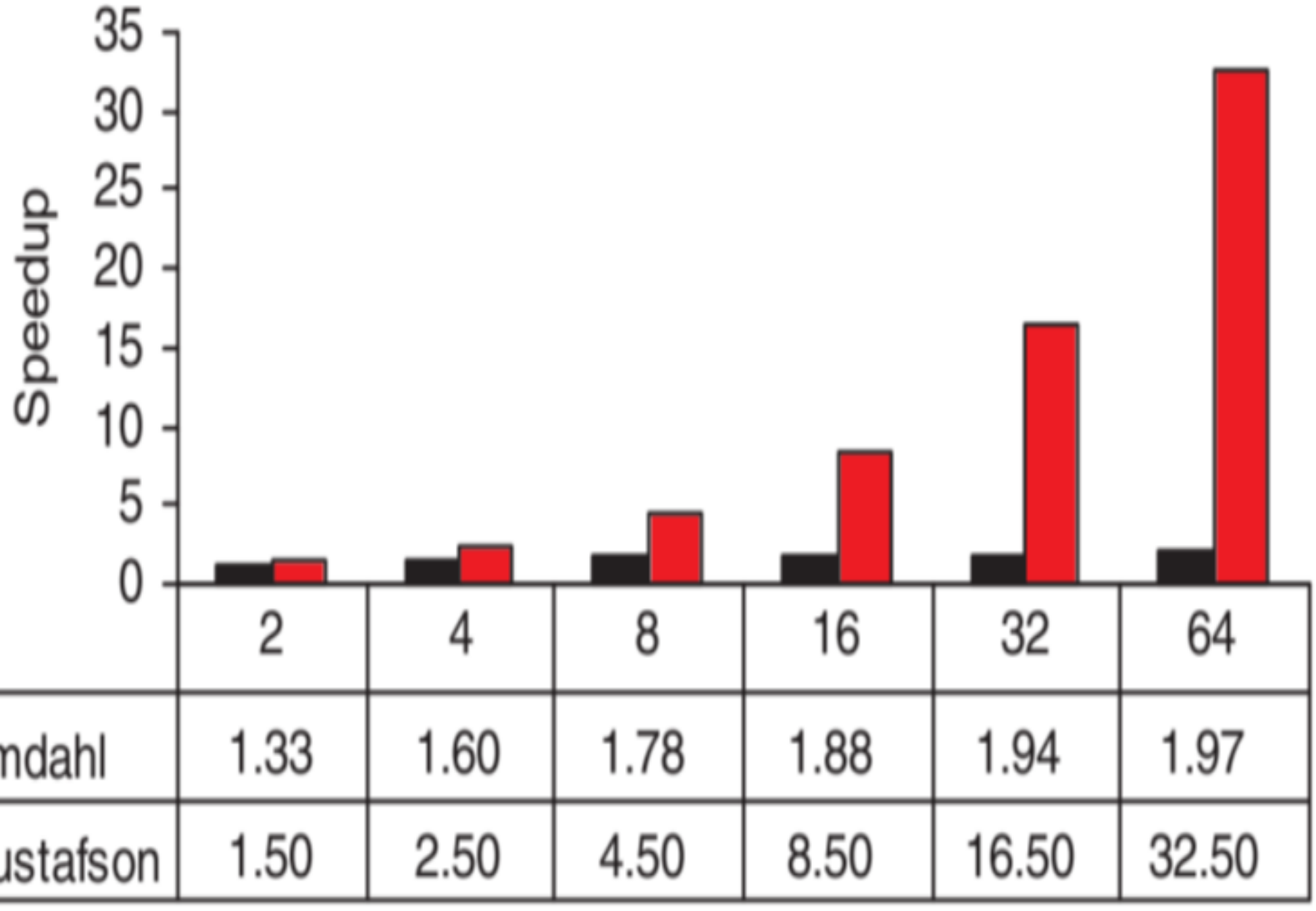
GUSTAFSON'S LAW

- Gustafson: Problem size generally scales with the number of processors or a more powerful processor rather than remaining constant.
  - The constant is the acceptable computing time.

$$Speedup_{Gustafson} = N - (n-1)S$$



THE LAWS COMPARED



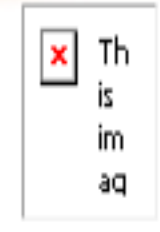
	2	4	8	16	32	64
Amdahl	1.33	1.60	1.78	1.88	1.94	1.97
Gustafson	1.50	2.50	4.50	8.50	16.50	32.50

HOW CAN WE DETERMINE IF COD
EXECUTES APPROPRIATELY?

- Logic Analyzer. - Oscilliscope
  No intrusion in the idea
- System clock → tools that measure time
- Instruction counting the assembly language code → Gcc.

WHAT IS PROFILING?

- Allows you to learn:
  - where your program is spending its time
  - what functions called what other functions
- Can show you which pieces of your program are slower than you expected
  - might be candidates for rewriting
- Are functions are being called more or less often than expected?



PROFILER STEPS

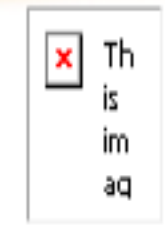
- Compile and link your program with profiling enabled
- Execute your program to generate a profile data file
- Run gprof to analyze the profile data

THE PROFILER

- Flat profiler
  - time your program spent in each function
  - how many times that function was called
  - information on which functions burn most of the cycles is clearly indicated here
- Call graph
  - shows, for each function:
    - which functions called it,
    - which other functions it called,
    - how many times it was called.

OUTPUT FROM

- an estimate of how much time was spent in the subroutines of each function
- suggests places where you might try to eliminate function calls that use a lot of time.



LETS LOOK AT AN EXAMPLE