Think Parallel

James Reinders

intel®

Possible topics (subject to change)

- How much parallelism is there? Where is it?
- 1024 chickens or a strong oxen?
- Observations based on what I hear and saw from you all
- Discuss results
- Ponder the future
  - What does it mean to program in the future?
  - Mechanics vs. Users?
- Q&A

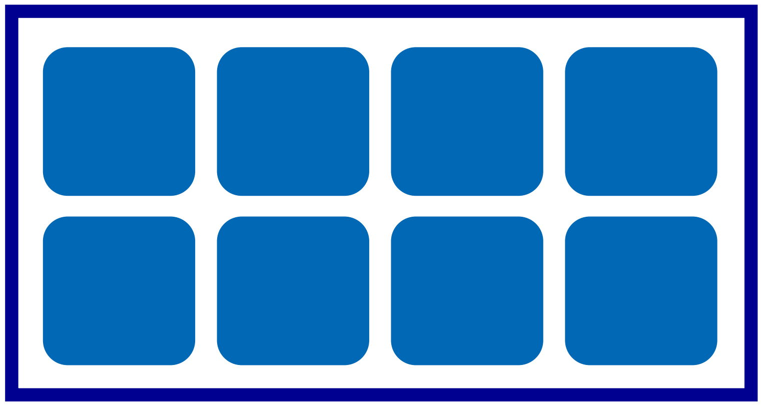
Possible topics (subject to change)

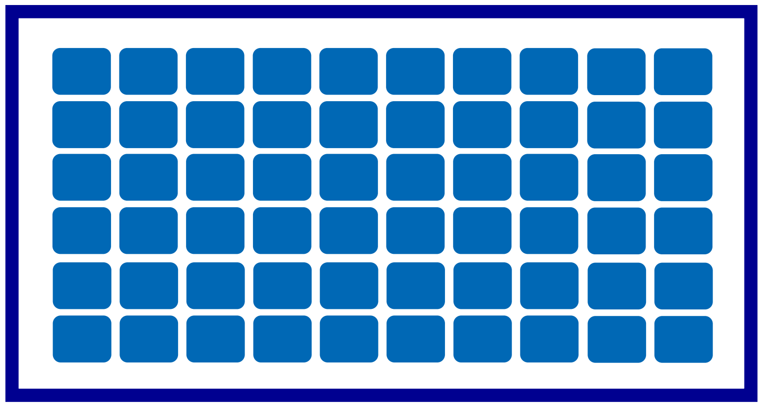
- How much parallelism is there? Where is it?
- 1024 chickens or a strong oxen?
- Observations based on what I hear and saw from you all
- Discuss results
- Ponder the future
  - What does it mean to program in the future?
  - Mechanics vs. Users?
- Q&A

How much parallelism is there?

- Amdahl’s Law
- Gustafson’s observations on Amdahl’s Law

Design Question: Computation?

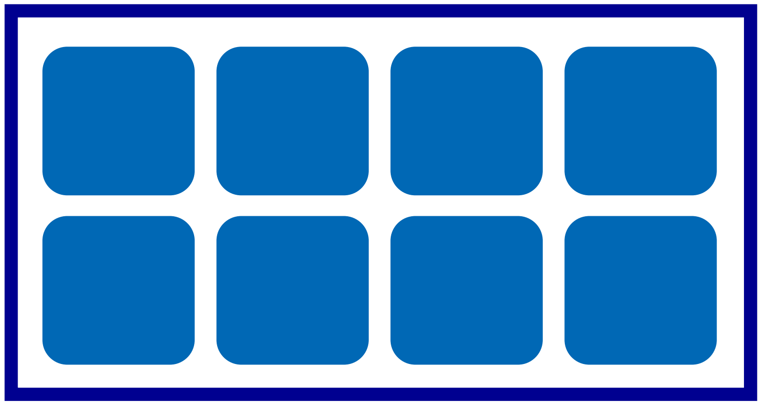


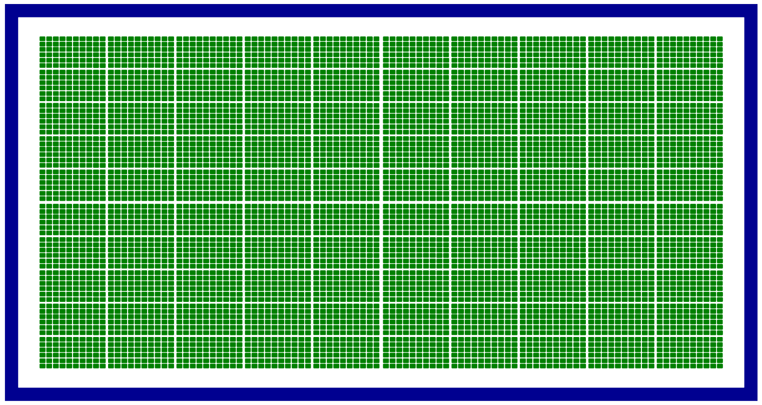


A few powerful vs. Many less powerful.

Diagrams for discussion purposes only, not a precise representation of any product of any company.

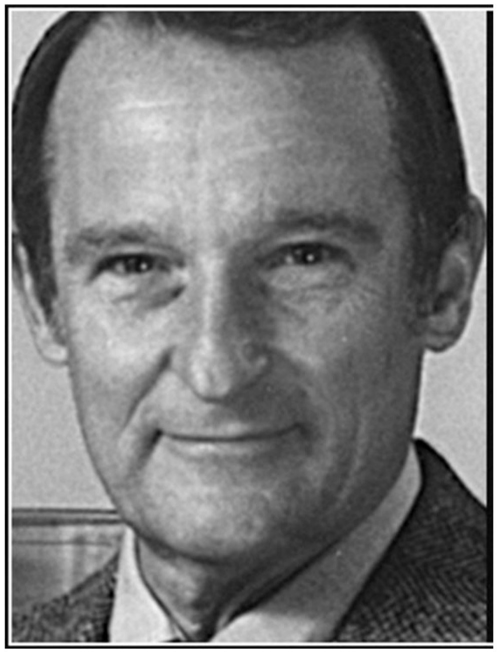
Design Question



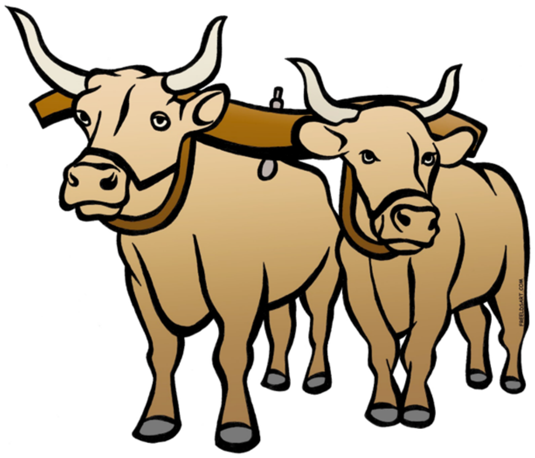


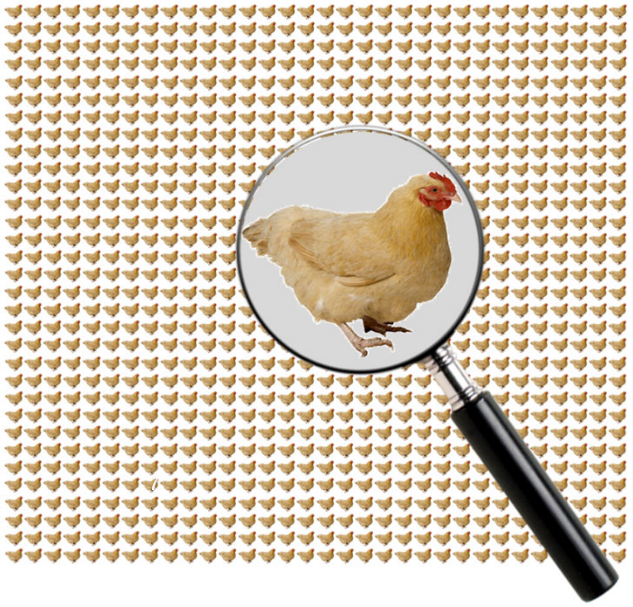
A few powerful *and* *very restrictive*. vs. Many *much* less powerful

Diagrams for discussion purposes only, not a precise representation of any product of any company.



If you were plowing a field,
which would you rather use…
two strong oxen, or
1024 chickens?

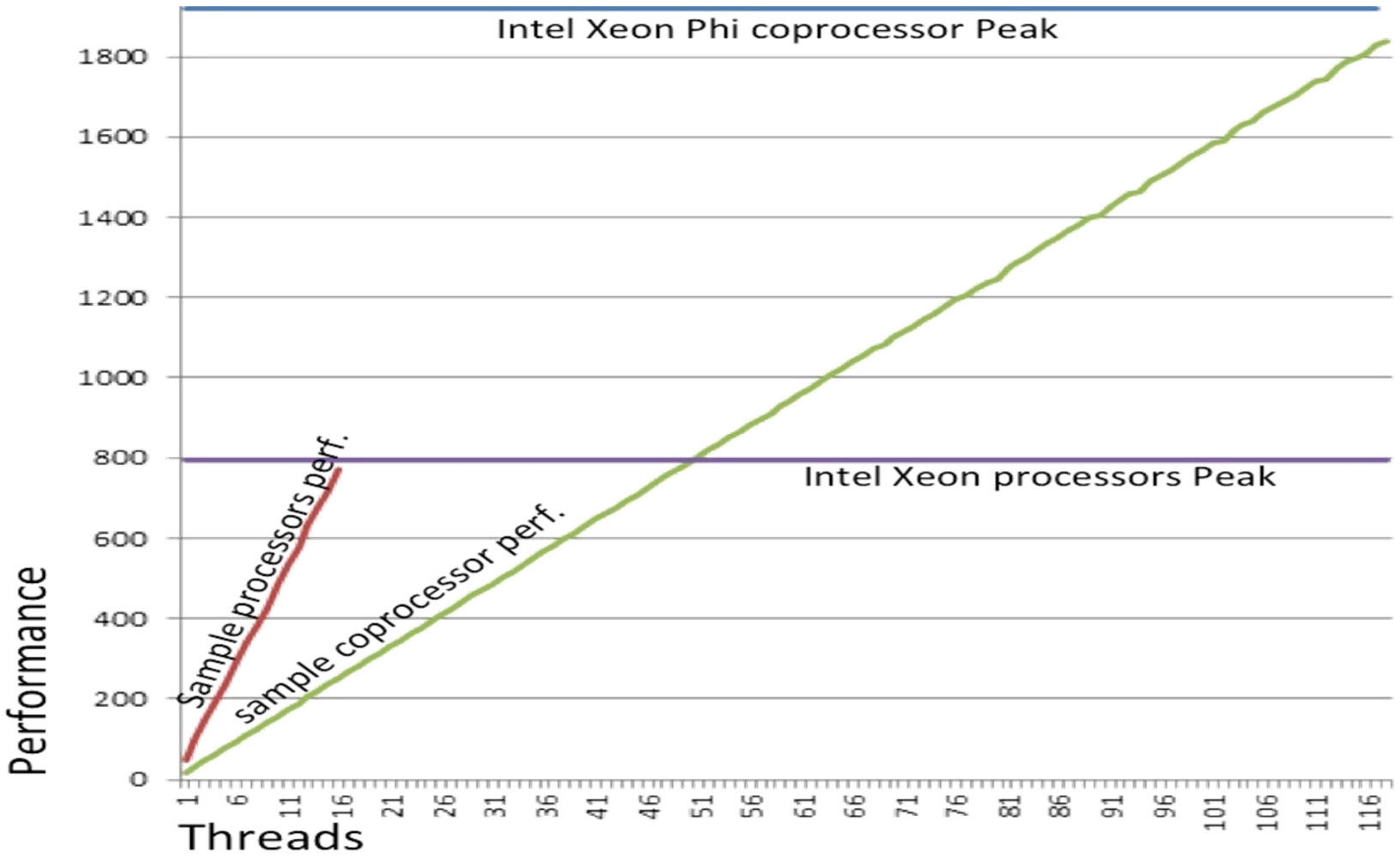




SCALE
(Go Parallel)

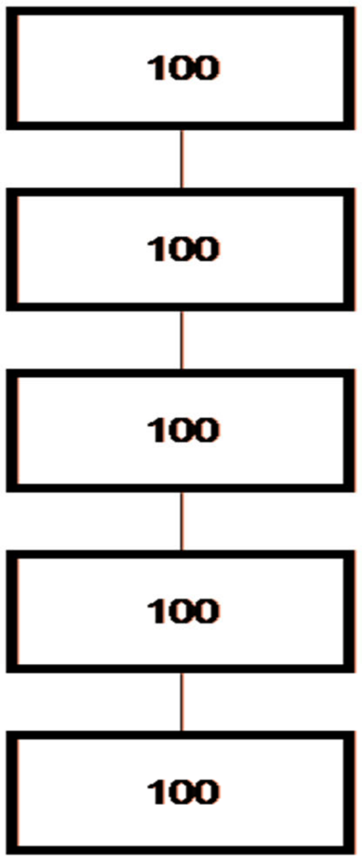
Eleven years ago. Old, but the reality never goes away.

Scaling can lead to higher peak even with lower ramp

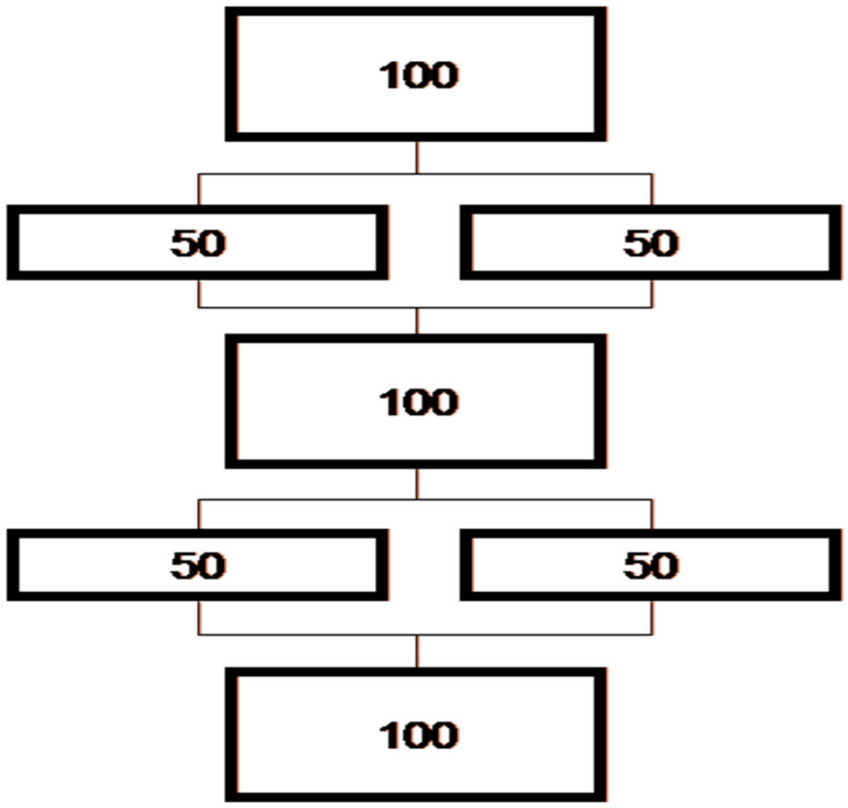


How much parallelism is there?

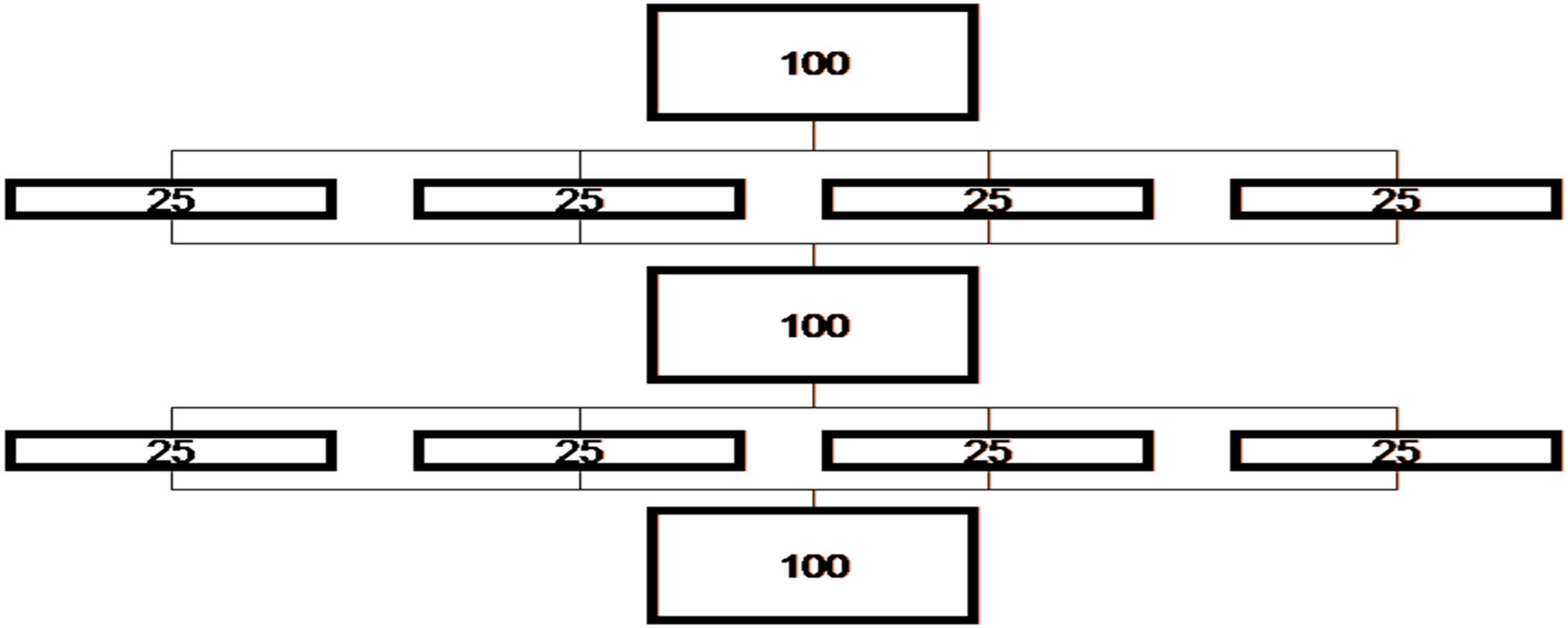
- Amdahl’s Law
- Gustafson’s observations on Amdahl’s Law



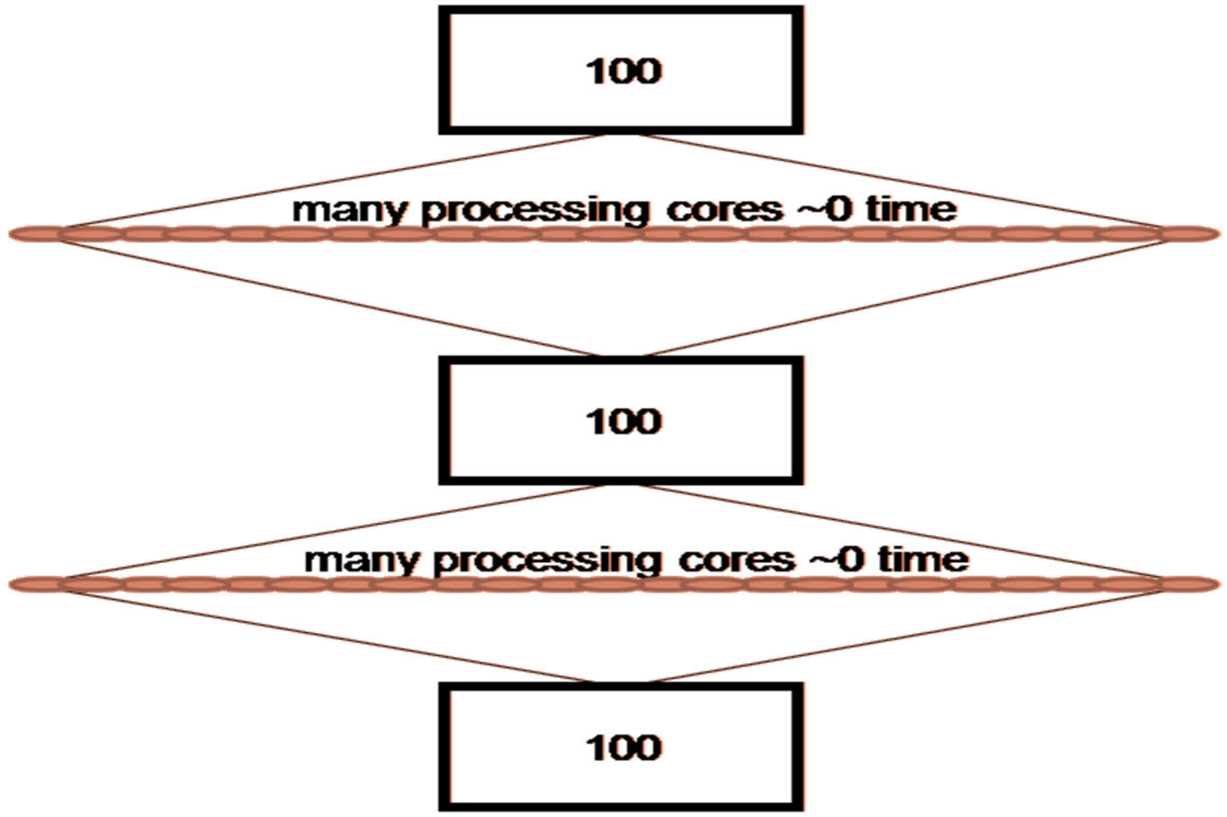
Work 500 Time 500
Speedup 1X



Work 500 Time 400
Speedup 1.25X



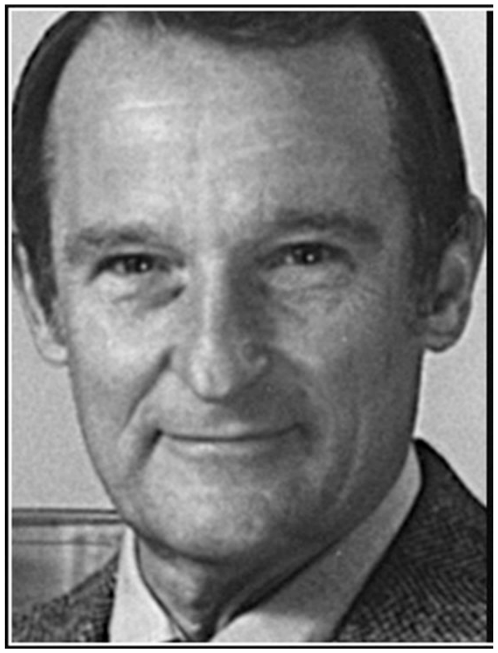
Work 500 Time 350
Speedup 1.4X



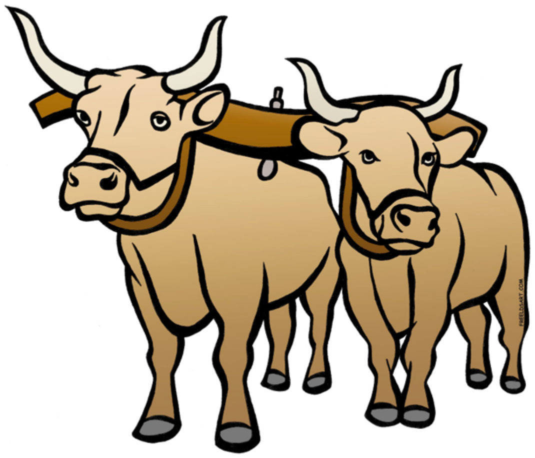
Work 500 Time 300
Speedup 1.7X

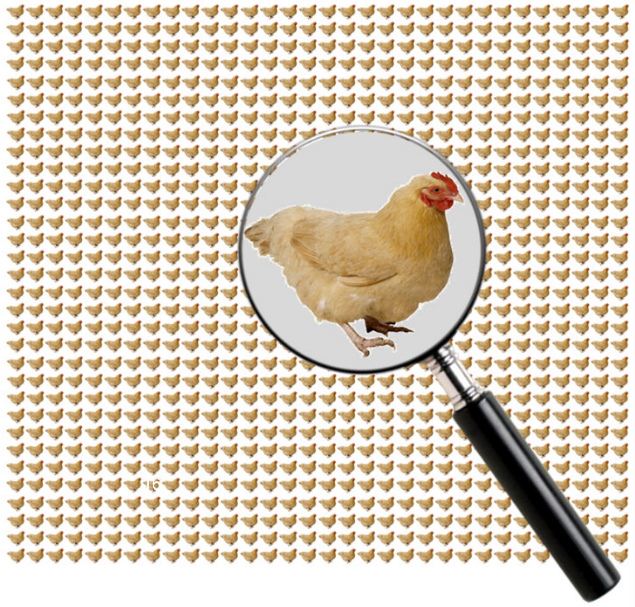
Amdahl's law

- "…the effort expended on achieving high parallel processing rates is wasted unless it is accompanied by achievements in sequential processing rates of very nearly the same magnitude."
  - – Amdahl, 1967



If you were plowing a field,
which would you rather use…
two strong oxen, or
1024 chickens?

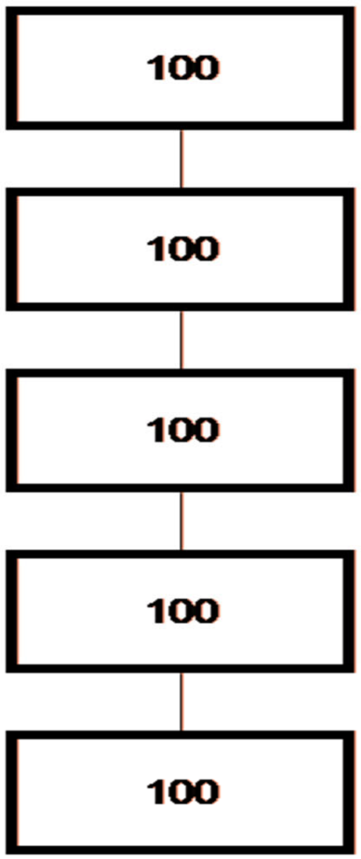




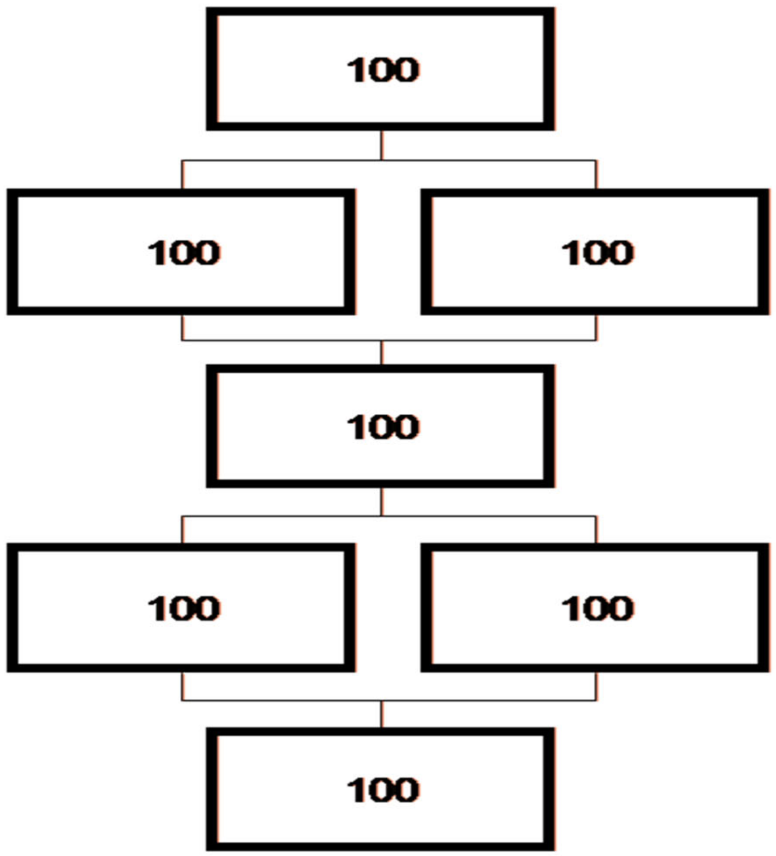
Amdahl’s law – an observation

- “…speedup should be measured by scaling the problem to the number of processors, not by fixing the problem size.”

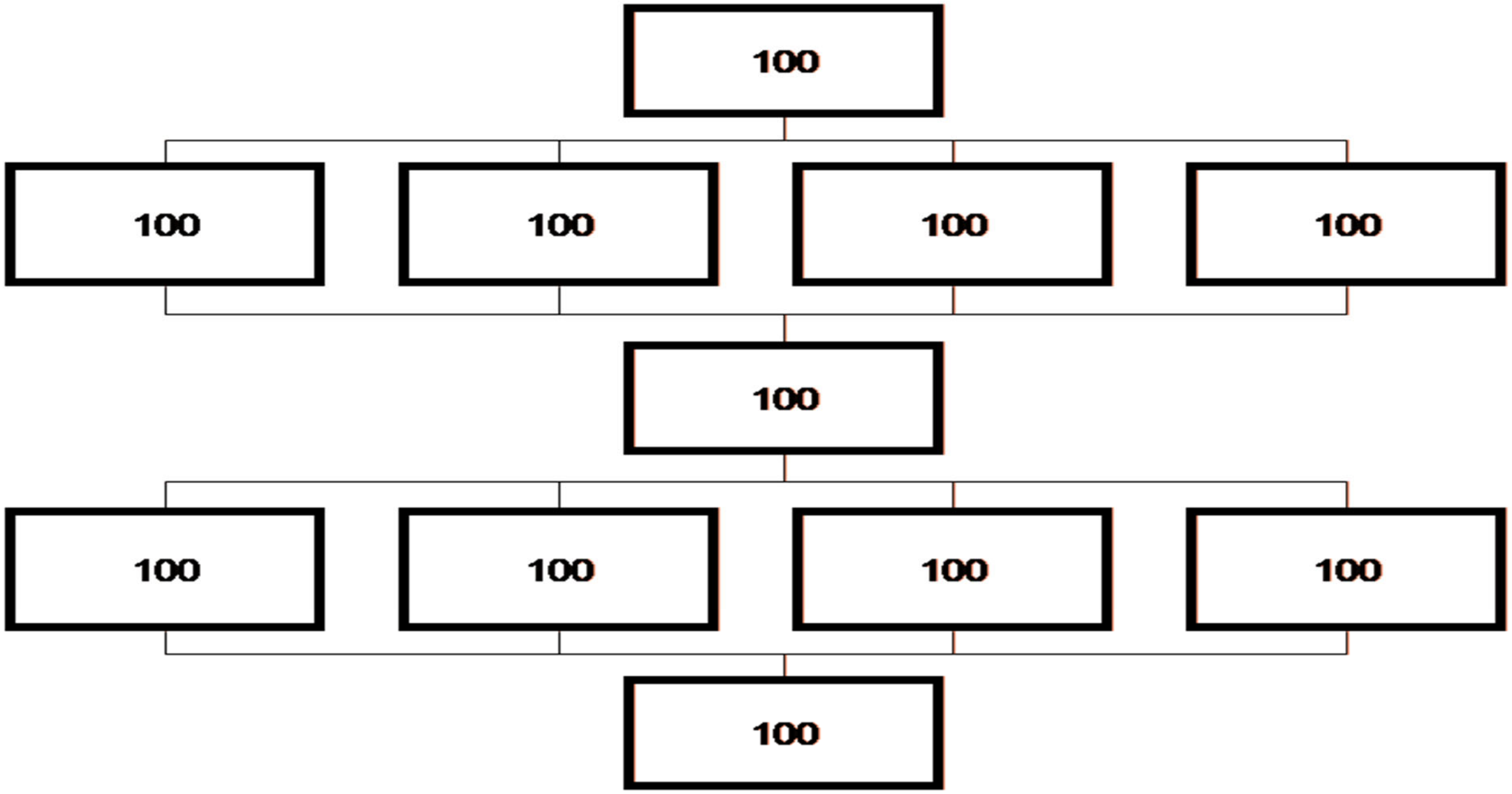
  - – Gustafson, 1988



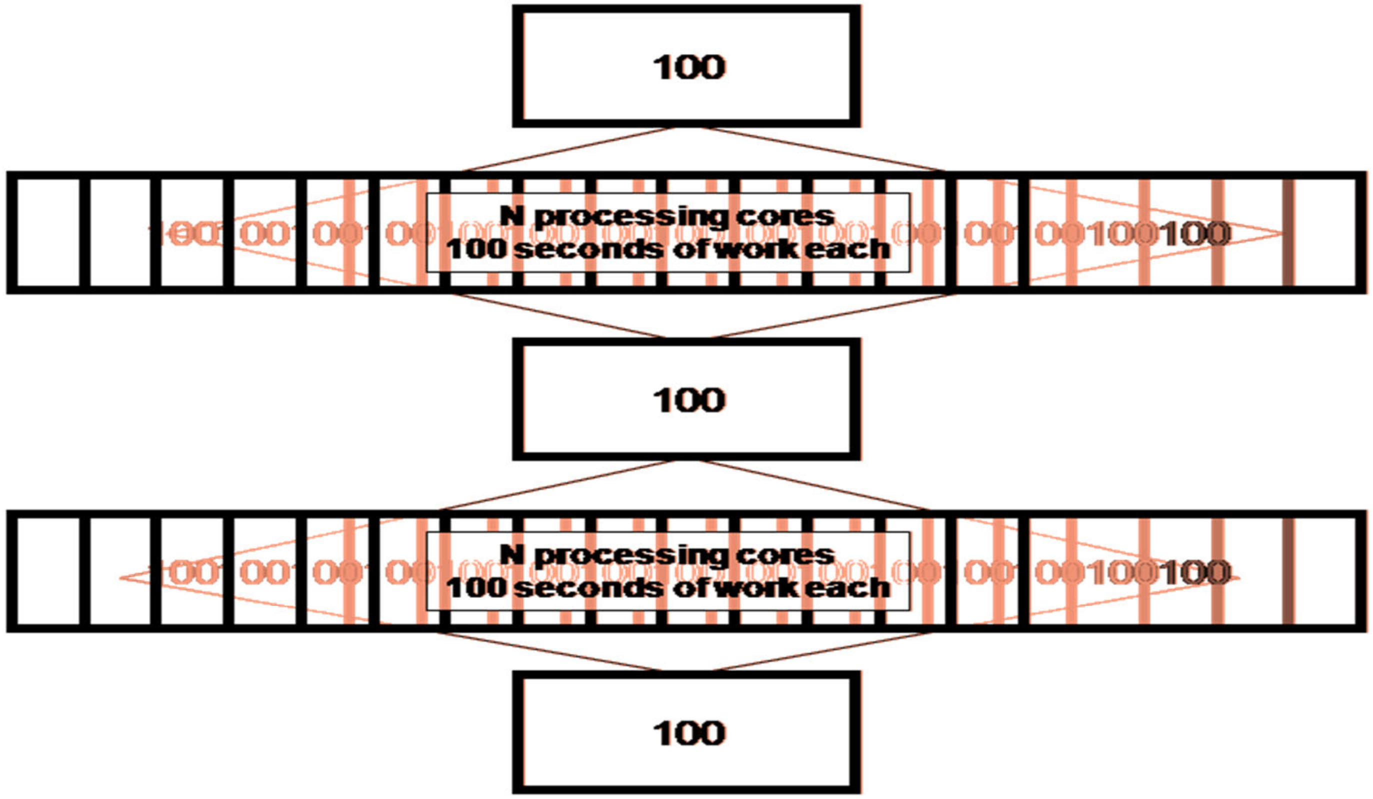
Work 500 Time 500
Speedup 1X



Work 700 Time 500
Speedup 1.4X



Work 1100 Time 500
Speedup 2.2X



Work 2*N*100+300 Time 500
Speedup O(N)

How much parallelism is there?

- Amdahl's Law
- Gustafson's observations on Amdahl's Law

- Plenty –
- but the workloads need to continue to grow !

“Weak” Scaling is the Norm.

Scales assuming data size grows.

“Strong” Scaling is much less common (and not infinite).

Scales just by adding more processing power.

Provocative thought

data parallelism is real,
task parallelism
might as well be a myth?

How much parallelism is there?

- Amdahl’s Law
- Gustafson’s observations on Amdahl’s Law

- Plenty – the workloads need to continue to grow (but they always do – don’t they?)

Possible topics (subject to change)

- How much parallelism is there? Where is it?
- 1024 chickens or a strong oxen?
- Observations based on what I hear and saw from you all
- Discuss results
- Ponder the future
    - What does it mean to program in the future?
    - Mechanics vs. Users?
- Q&A

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
  - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)
- Observation critical
    - In debugging: the unexpected can tell you more than the expected
    - First step: know what you expect BEFORE you see results
    - Careful timing important

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)
- Observation critical
    - In debugging: the unexpected can tell you more than the expected
    - First step: know what you expect BEFORE you see results
    - Careful timing important
- Warming caches, pipelines, etc.

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)
- Observation critical
    - In debugging: the unexpected can tell you more than the expected
    - First step: know what you expect BEFORE you see results
    - Careful timing important
- Warming caches, pipelines, etc.
- What is under the covers? Do you really know?
    - The more you UNDERSTAND what you are programming – the more you can do

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)
- Observation critical
    - In debugging: the unexpected can tell you more than the expected
    - First step: know what you expect BEFORE you see results
    - Careful timing important
- Warming caches, pipelines, etc.
- What is under the covers? Do you really know?
    - The more you UNDERSTAND what you are programming – the more you can do
- Compare timings (THINK)

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)
- Observation critical
    - In debugging: the unexpected can tell you more than the expected
    - First step: know what you expect BEFORE you see results
    - Careful timing important
- Warming caches, pipelines, etc.
- What is under the covers? Do you really know?
    - The more you UNDERSTAND what you are programming – the more you can do
- Compare timings (THINK)
- Check results

Observations based on what I hear and saw from you all

- Unbalanced – keeping pi computation
    - A useful debug technique for parallel applications
- Amdahl/Gustafson - Having enough work (bigger images, etc.)
- Observation critical
    - In debugging: the unexpected can tell you more than the expected
    - First step: know what you expect BEFORE you see results
    - Careful timing important
- Warming caches, pipelines, etc.
- What is under the covers? Do you really know?
    - The more you UNDERSTAND what you are programming – the more you can do
- Compare timings (THINK)
- Check results
- Tell us what else you saw

Possible topics (subject to change)

- How much parallelism is there? Where is it?
- 1024 chickens or a strong oxen?
- Observations based on what I hear and saw from you all
- Discuss results
- Ponder the future
  - What does it mean to program in the future?
  - Mechanics vs. Users?
- Q&A

Thank you