Amdahl's and Gustafson's laws

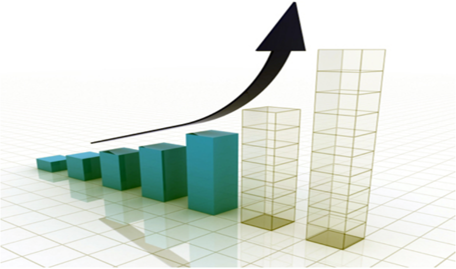
Jan Zapletal

VŠB - Technical University of Ostrava
jan.zapletal@vsb.cz

November 23, 2009

Performance analysis

How does the parallelization improve the performance of our program?

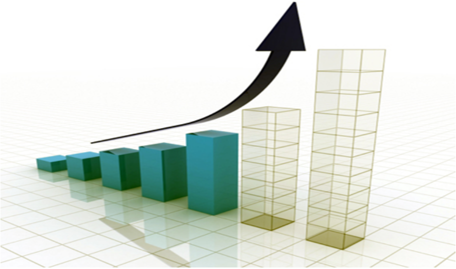


Metrics used to desribe the performance:

- execution time,
- speedup,
- efficiency,
- cost...

Performance analysis

How does the parallelization improve the performance of our program?



Metrics used to desribe the performance:

- execution time,
- speedup,
- efficiency,
- cost...

Metrics

Execution time

- The time elapsed from when the first processor starts the execution to when the last processor completes it.
- On a parallel system consists of computation time, communication time and idle time.

Speedup

- Defined as

$$S = \frac{T_1}{T_p},$$

where T_1 is the execution time for a sequential system and T_p for the parallel system.

Metrics

Execution time

- The time elapsed from when the first processor starts the execution to when the last processor completes it.
- On a parallel system consists of computation time, communication time and idle time.

Speedup

- Defined as
$$S = \frac{T_1}{T_p},$$
where T_1 is the execution time for a sequential system and T_p for the parallel system.

Amdahl's law

Gene Myron Amdahl (born November 16, 1922)

- worked for IBM,
- best known for formulating Amdahl's law uncovering the *limits of parallel computing*.

Let T_1 denote the computation time on a sequential system. We can split the total time as follows

$$T_1 = t_s + t_p,$$

where

- t_s - computation time needed for the sequential part.
- t_p - computation time needed for the parallel part.

Clearly, if we parallelize the problem, only t_p can be reduced. Assuming *ideal* parallelization we get

$$T_p = t_s + \frac{t_p}{N},$$

where

- N - number of processors.

Amdahl's law

Gene Myron Amdahl (born November 16, 1922)

- worked for IBM,
- best known for formulating Amdahl's law uncovering the *limits of parallel computing*.

Let T_1 denote the computation time on a sequential system. We can split the total time as follows

$$T_1 = t_s + t_p,$$

where

- t_s - computation time needed for the sequential part.
- t_p - computation time needed for the parallel part.

Clearly, if we parallelize the problem, only t_p can be reduced. Assuming *ideal* parallelization we get

$$T_p = t_s + \frac{t_p}{N},$$

where

- N - number of processors.

Amdahl's law

Gene Myron Amdahl (born November 16, 1922)

- worked for IBM,
- best known for formulating Amdahl's law uncovering the *limits of parallel computing*.

Let T_1 denote the computation time on a sequential system. We can split the total time as follows

$$T_1 = t_s + t_p,$$

where

- t_s - computation time needed for the sequential part.
- t_p - computation time needed for the parallel part.

Clearly, if we parallelize the problem, only t_p can be reduced. Assuming *ideal* parallelization we get

$$T_p = t_s + \frac{t_p}{N},$$

where

- N - number of processors.

Amdahl's law

Thus we get the speedup of

$$S = \frac{T_1}{T_p} = \frac{t_s + t_p}{t_s + \frac{t_p}{N}}.$$

Let f denote the sequential portion of the computation, i.e.

$$f = \frac{t_s}{t_s + t_p}.$$

Thus the speedup formula can be simplified into

$$S = \frac{1}{f + \frac{1-f}{N}} < \frac{1}{f}.$$

- Notice that Amdahl assumes the problem size does not change with the number of CPUs.
- Wants to solve a fixed-size problem as quickly as possible.

Amdahl's law

Thus we get the speedup of

$$S = \frac{T_1}{T_p} = \frac{t_s + t_p}{t_s + \frac{t_p}{N}}.$$

Let f denote the sequential portion of the computation, i.e.

$$f = \frac{t_s}{t_s + t_p}.$$

Thus the speedup formula can be simplified into

$$S = \frac{1}{f + \frac{1-f}{N}} < \frac{1}{f}.$$

- Notice that Amdahl assumes the problem size does not change with the number of CPUs.
- Wants to solve a fixed-size problem as quickly as possible.

Amdahl's law

Thus we get the speedup of

$$S = \frac{T_1}{T_p} = \frac{t_s + t_p}{t_s + \frac{t_p}{N}}.$$

Let f denote the sequential portion of the computation, i.e.

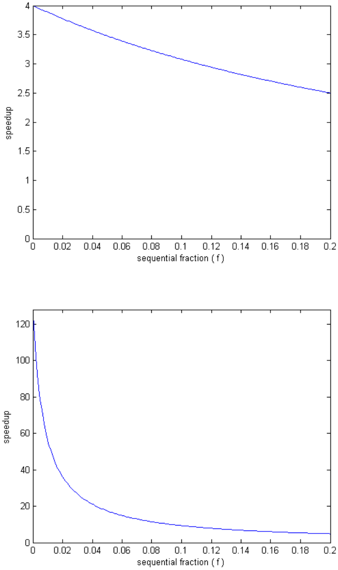
$$f = \frac{t_s}{t_s + t_p}.$$

Thus the speedup formula can be simplified into

$$S = \frac{1}{f + \frac{1-f}{N}} < \frac{1}{f}.$$

- Notice that Amdahl assumes the problem size does not change with the number of CPUs.
- Wants to solve a fixed-size problem as quickly as possible.

Amdahl's law



speedup
sequential fraction (f)
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
0
0.5
1
1.5
2
2.5
3
3.5
4
2
4
6
8
10
12
14
16
20
40
60
80
100
120
100
200
300
400
500
600
700
800
900
1000

Gustafson's law

John L. Gustafson (born January 19, 1955)

- American computer scientist and businessman,
- found out that practital problems show much better speedup than Amdahl predicted.

Gustafson's law

- The computation time is constant (instead of the problem size),
- increasing number of CPUs ⇒ solve bigger problem and get better results *in the same time*.

Let T_p denote the computation time on a parallel system. We can split the total time as follows

$$T_p = t_s^* + t_p^*,$$

where

- t_s^* - computation time needed for the sequential part.
- t_p^* - computation time needed for the parallel part.

Gustafson's law

John L. Gustafson (born January 19, 1955)

- American computer scientist and businessman,
- found out that practital problems show much better speedup than Amdahl predicted.

Gustafson's law

- The computation time is constant (instead of the problem size),
- increasing number of CPUs $\Rightarrow$ solve bigger problem and get better results *in the same time*.

Let T_p denote the computation time on a parallel system. We can split the total time as follows

$$T_p = t_s^* + t_p^*,$$

where

- t_s^* - computation time needed for the sequential part.
- t_p^* - computation time needed for the parallel part.

Gustafson's law

John L. Gustafson (born January 19, 1955)

- American computer scientist and businessman,
- found out that practital problems show much better speedup than Amdahl predicted.

Gustafson's law

- The computation time is constant (instead of the problem size),
- increasing number of CPUs $\Rightarrow$ solve bigger problem and get better results *in the same time*.

Let T_p denote the computation time on a parallel system. We can split the total time as follows

$$T_p = t_s^* + t_p^*,$$

where

- t_s^* - computation time needed for the sequential part.
- t_p^* - computation time needed for the parallel part.

Gustafson's law

On a sequential system we would get

$$T_1 = t_s^* + N \cdot t_p^*.$$

Thus the speedup will be

$$S = \frac{t_s^* + N \cdot t_p^*}{t_s^* + t_p^*}.$$

Let f^* denote the sequential portion of the computation *on the parallel system*, i.e.

$$f^* = \frac{t_s^*}{t_s^* + t_p^*}.$$

Then

$$S = f^* + N \cdot (1 - f^*).$$

Gustafson's law

On a sequential system we would get

$$T_1 = t_s^* + N \cdot t_p^*.$$

Thus the speedup will be

$$S = \frac{t_s^* + N \cdot t_p^*}{t_s^* + t_p^*}.$$

Let f^* denote the sequential portion of the computation *on the parallel system*, i.e.

$$f^* = \frac{t_s^*}{t_s^* + t_p^*}.$$

Then

$$S = f^* + N \cdot (1 - f^*).$$

Gustafson's law

On a sequential system we would get

$$T_1 = t_s^* + N \cdot t_p^*.$$

Thus the speedup will be

$$S = \frac{t_s^* + N \cdot t_p^*}{t_s^* + t_p^*}.$$

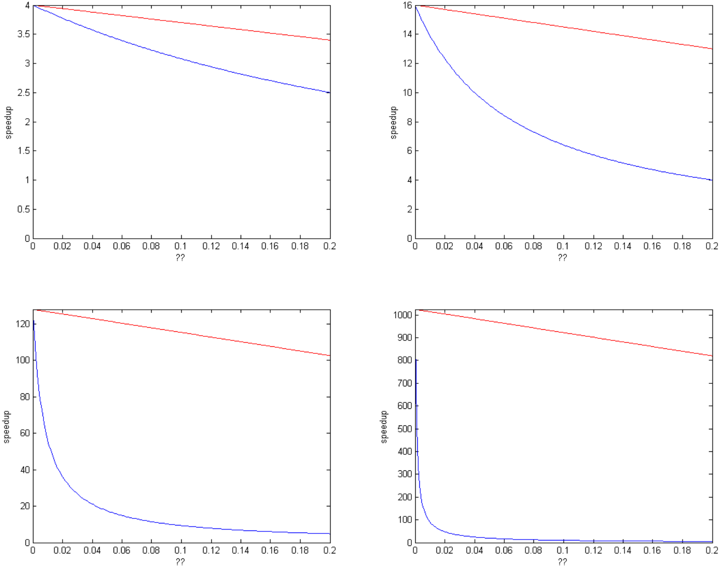
Let f^* denote the sequential portion of the computation *on the parallel system*, i.e.

$$f^* = \frac{t_s^*}{t_s^* + t_p^*}.$$

Then

$$S = f^* + N \cdot (1 - f^*).$$

Gustafson's law



What the hell?!

- The bigger the problem, the smaller f - serial part remains usualy the same,
- and $f \neq f^*$.

Amdahl's says:

$$S = \frac{t_s + t_p}{t_s + \frac{t_p}{N}}.$$

Let now f^* denote the sequential portion spent in the parallel computation, i.e.

$$f^* = \frac{t_s}{t_s + \frac{t_p}{N}} \text{ and } (1 - f^*) = \frac{\frac{t_p}{N}}{t_s + \frac{t_p}{N}}.$$

Hence

$$t_s = f^* \cdot \left(t_s + \frac{t_p}{N}\right) \text{ and } t_p = N \cdot (1 - f^*) \cdot \left(t_s + \frac{t_p}{N}\right).$$

What the hell?!

- The bigger the problem, the smaller f - serial part remains usualy the same,
- and $f \neq f^*$.

Amdahl's says:

$$S = \frac{t_s + t_p}{t_s + \frac{t_p}{N}}.$$

Let now f^* denote the sequential portion spent in the parallel computation, i.e.

$$f^* = \frac{t_s}{t_s + \frac{t_p}{N}} \text{ and } (1 - f^*) = \frac{\frac{t_p}{N}}{t_s + \frac{t_p}{N}}.$$

Hence

$$t_s = f^* \cdot \left(t_s + \frac{t_p}{N}\right) \text{ and } t_p = N \cdot (1 - f^*) \cdot \left(t_s + \frac{t_p}{N}\right).$$

What the hell?!

- The bigger the problem, the smaller f - serial part remains usualy the same,
- and $f \neq f^*$.

Amdahl's says:

$$S = \frac{t_s + t_p}{t_s + \frac{t_p}{N}}.$$

Let now f^* denote the sequential portion spent in the parallel computation, i.e.

$$f^* = \frac{t_s}{t_s + \frac{t_p}{N}} \text{ and } (1 - f^*) = \frac{\frac{t_p}{N}}{t_s + \frac{t_p}{N}}.$$

Hence

$$t_s = f^* \cdot \left(t_s + \frac{t_p}{N}\right) \text{ and } t_p = N \cdot (1 - f^*) \cdot \left(t_s + \frac{t_p}{N}\right).$$

I see!

- After substituting t_s and t_p into the Amdahl's formula one gets

$$S = \frac{t_s + t_p}{t_s + \frac{t_p}{N}} = f^* + N \cdot (1 - f^*),$$

  what is exactly what Gustafson derived.
- The key is not to mix up the values f and f^* - this caused great confusion that lasted over years!

I see!

- After substituting t_s and t_p into the Amdahl's formula one gets

$$S = \frac{t_s + t_p}{t_s + \frac{t_p}{N}} = f^* + N \cdot (1 - f^*),$$

  what is exactly what Gustafson derived.
- The key is not to mix up the values f and f^* - this caused great confusion that lasted over years!

I see!

- After substituting t_s and t_p into the Amdahl's formula one gets

$$S = \frac{t_s + t_p}{t_s + \frac{t_p}{N}} = f^* + N \cdot (1 - f^*),$$

  what is exactly what Gustafson derived.

- The key is not to mix up the values f and f^* - this caused great confusion that lasted over years!

References

- QUINN, Michael Jay. *Parallel programming in C with MPI and OpenMP.* New York : McGraw - Hill, 2004. 507 s.
- *Amdahl's law [online].* Available at: <http://en.wikipedia.org/wiki/Amdahl's_law>.
- *Gustafson's law [online].* Available at: <http://en.wikipedia.org/wiki/Gustafson's_law>.

Thank you for your attention!

References

QUINN, Michael Jay. *Parallel programming in C with MPI and OpenMP.* New York : McGraw - Hill, 2004. 507 s.

Amdahl's law [online]. Available at: <http://en.wikipedia.org/wiki/Amdahl's_law>.

Gustafson's law [online]. Available at: <http://en.wikipedia.org/wiki/Gustafson's_law>.

Thank you for your attention!