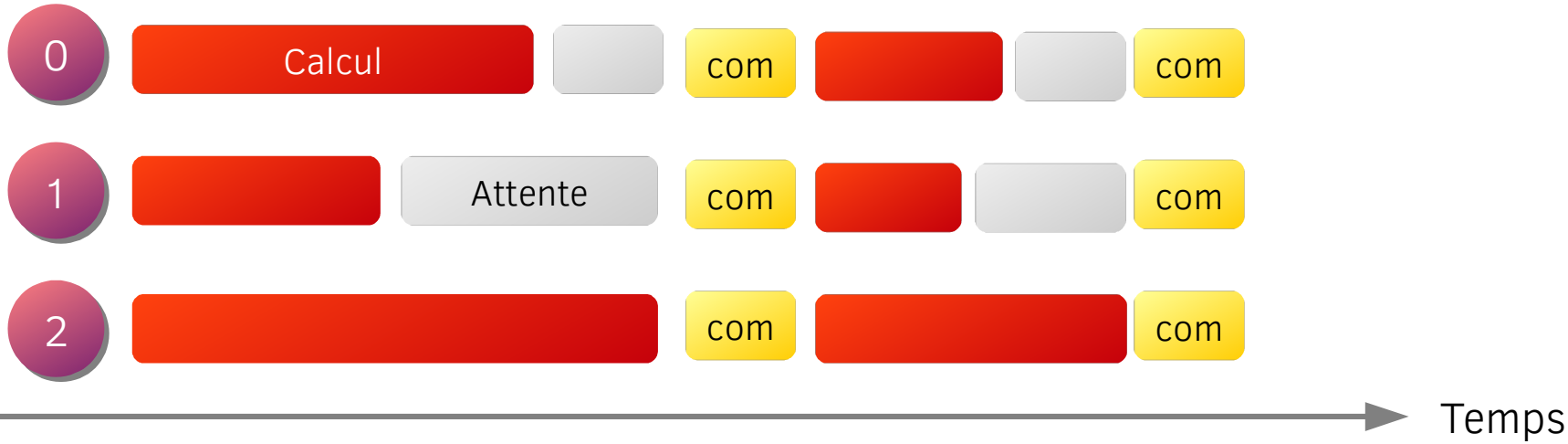
IV. Mesure de la Performance

Déséquilibre de charge

Load imbalance

On parle de déséquilibre de charge lorsque chaque processus MPI d'une même exécution n'a pas la même quantité de travail (calcul par exemple) à effectuer entraînant des différences de temps de traitement et par conséquent des phases d'attente non désirées.



De l'attente n'est ni plus ni moins que de la ressource de calcul gaspillée.

Parallélisme parfait

Embarrassingly parallel

Le parallélisme parfait est un parallélisme qui ne contient aucune zone séquentielle, aucun échange, aucune dépendance, aucune synchronisation entre les processus.

Scalabilité ou passage à l'échelle

Scalability

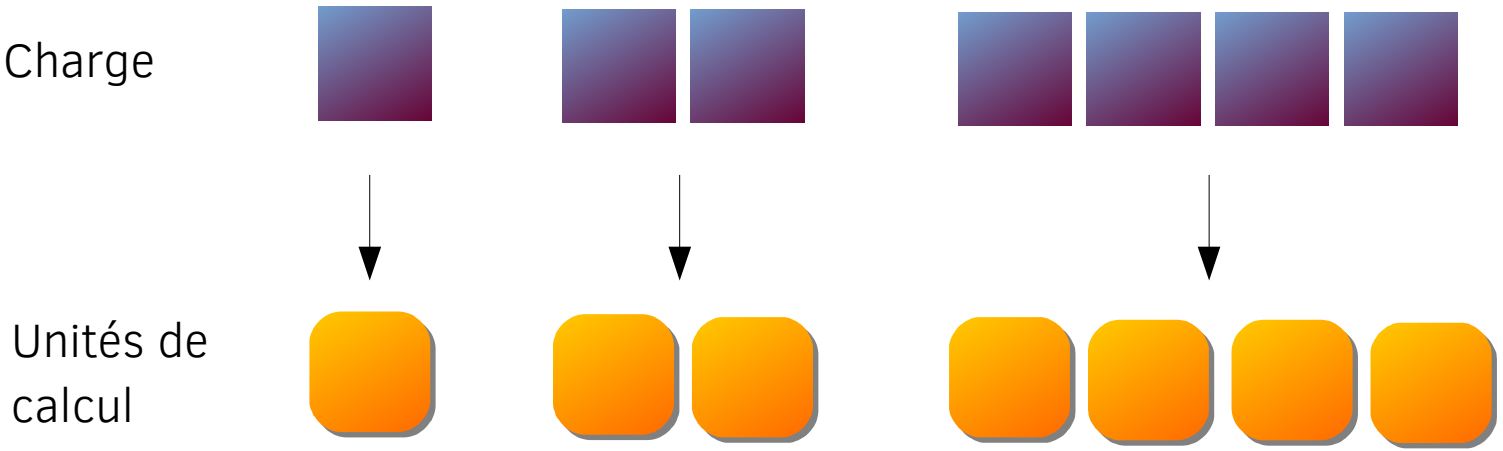
La scalabilité ou la capacité de passage à l'échelle donne la capacité d'un programme parallèle à s'adapter à l'augmentation du nombre d'unités de calcul sur un système donné.

- Cette étude est systématique pour déterminer le potentiel d'un code parallèle
- Cette étude se présente sous la forme d'un graphe
- Elle permet de mesurer le nombre maximal d'unités de calcul utiles pour la résolution d'un problème

Scalabilité ou passage à l'échelle faible

Weak Scalability

Une étude de scalabilité faible consiste à garder la charge constante par processus tout en augmentant le nombre de processus. Un processus tourne sur une seule unité de calcul.

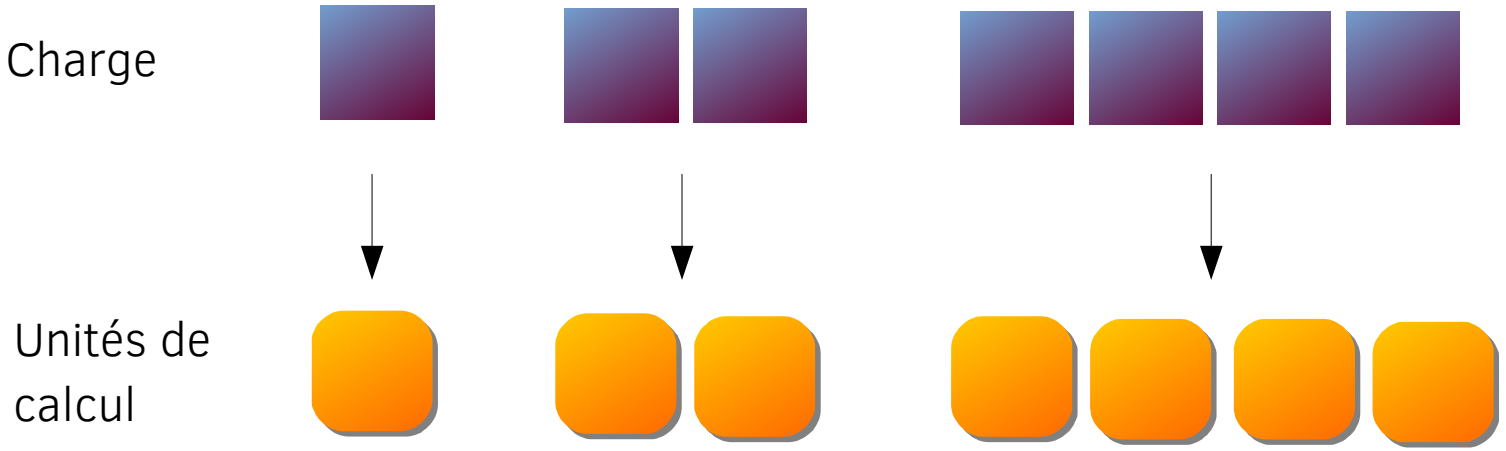


https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

Scalabilité ou passage à l'échelle faible

Weak Scalability

La taille total du domaine augmente proportionnellement au nombre d'unités de calcul. Le nombre de communications/synchronisations augmente lui aussi de la même manière.



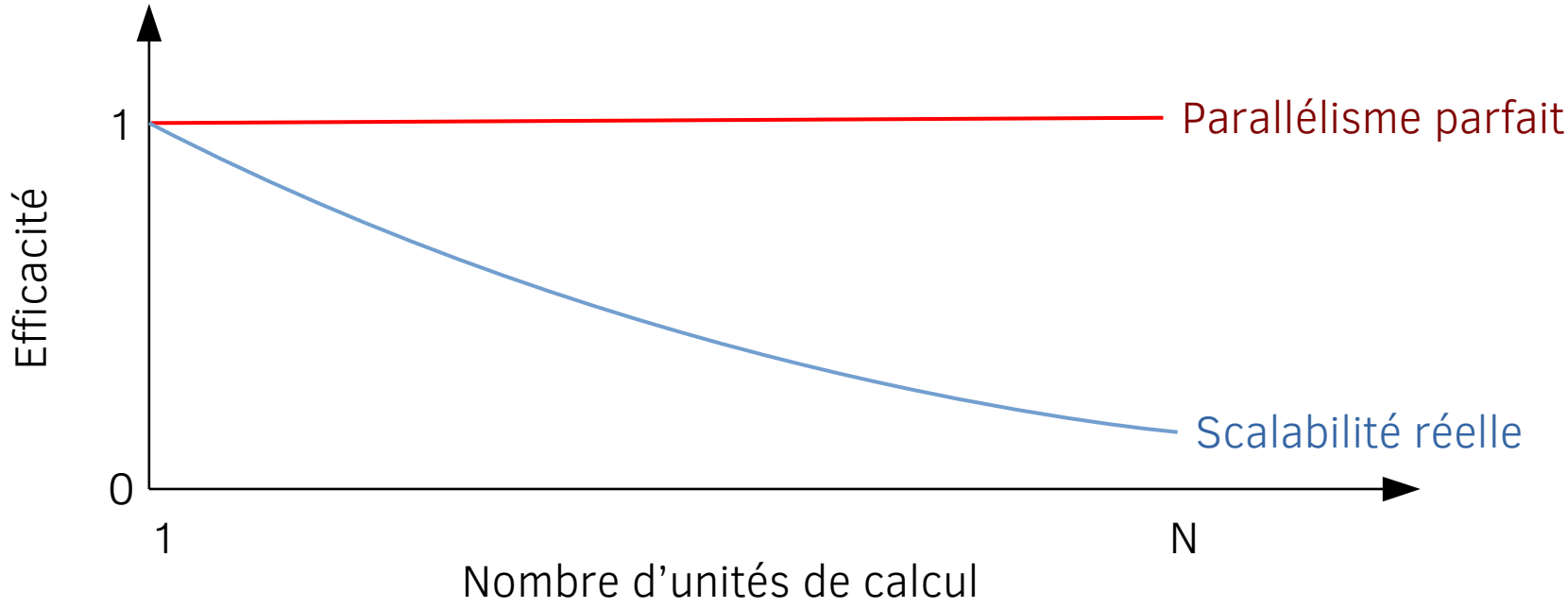
https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

Scalabilité ou passage à l'échelle faible

Weak Scalability

On trace généralement l'efficacité de scalabilité faible :
t(1) est le temps de simulation pou un processus
t(N) est le temps de simulation pour N processus

$$E_{faible}(N) = \frac{t(1)}{t(N)}$$

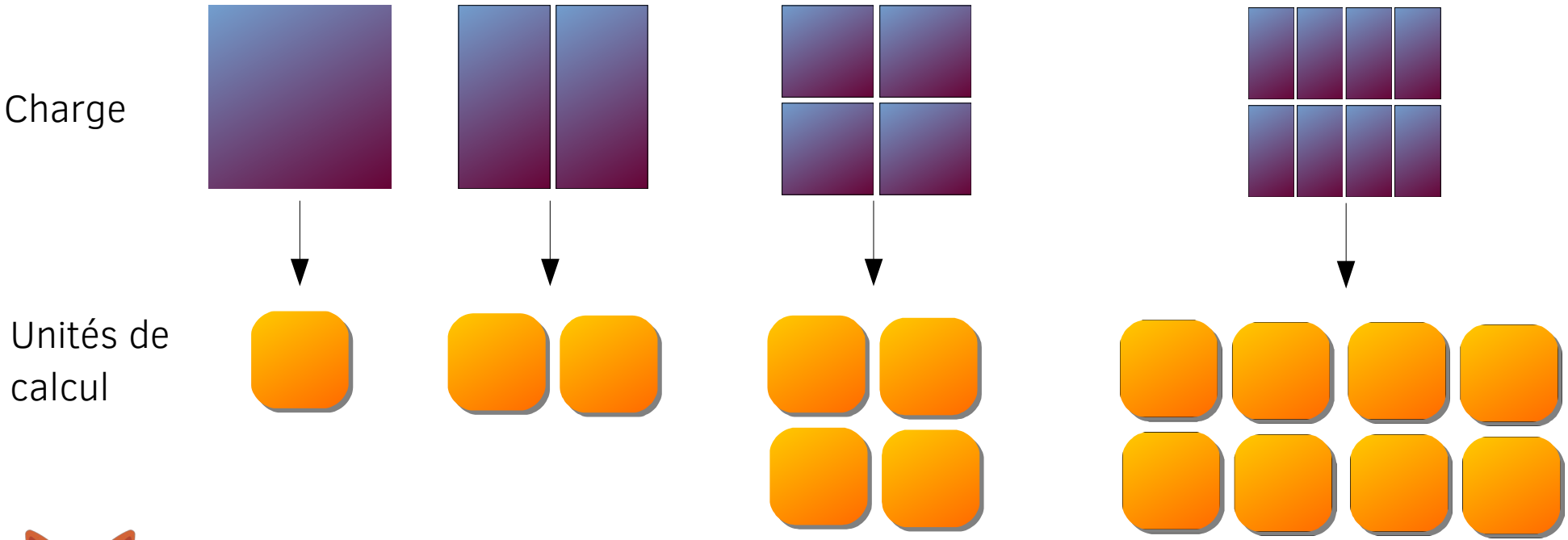


https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

Scalabilité ou passage à l'échelle forte

Strong Scalability

Une étude de scalabilité forte consiste à garder la charge totale constante tout en augmentant le nombre de processus. La charge par processus varie donc en fonction du nombre de processus.

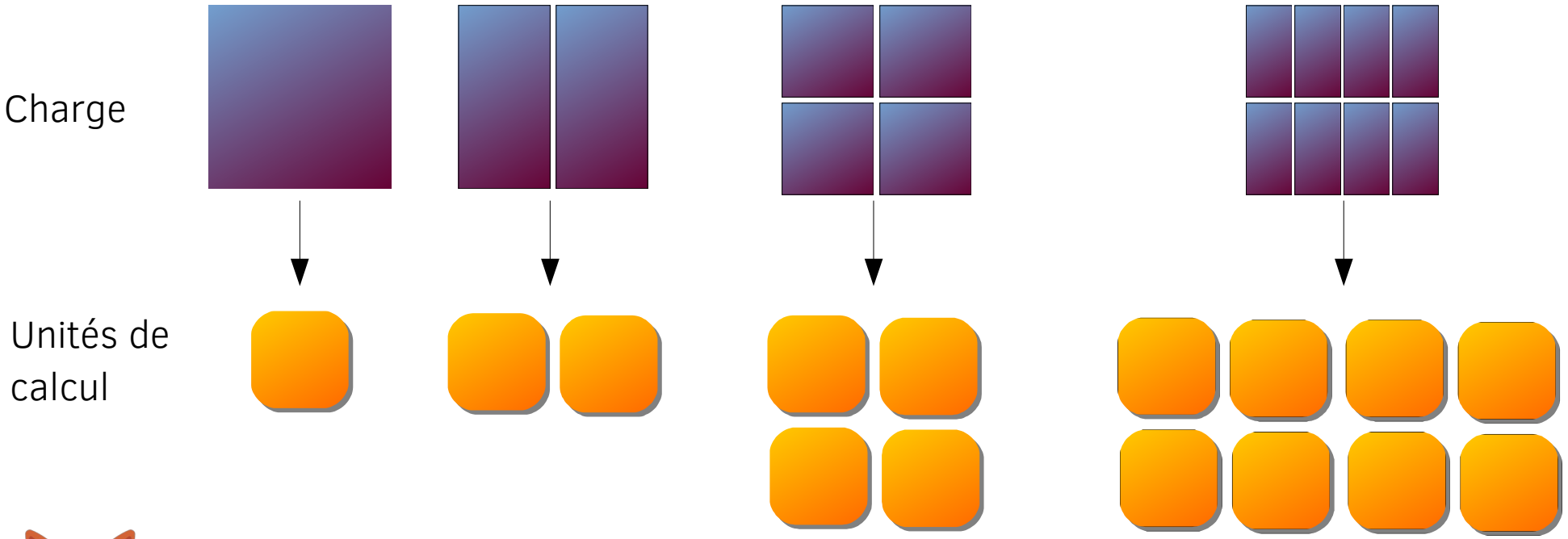


https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

Scalabilité ou passage à l'échelle forte

Strong Scalability

Le domaine garde sa taille constante mais le nombre de communications et de synchronisations augmente avec le nombre de subdivisions.



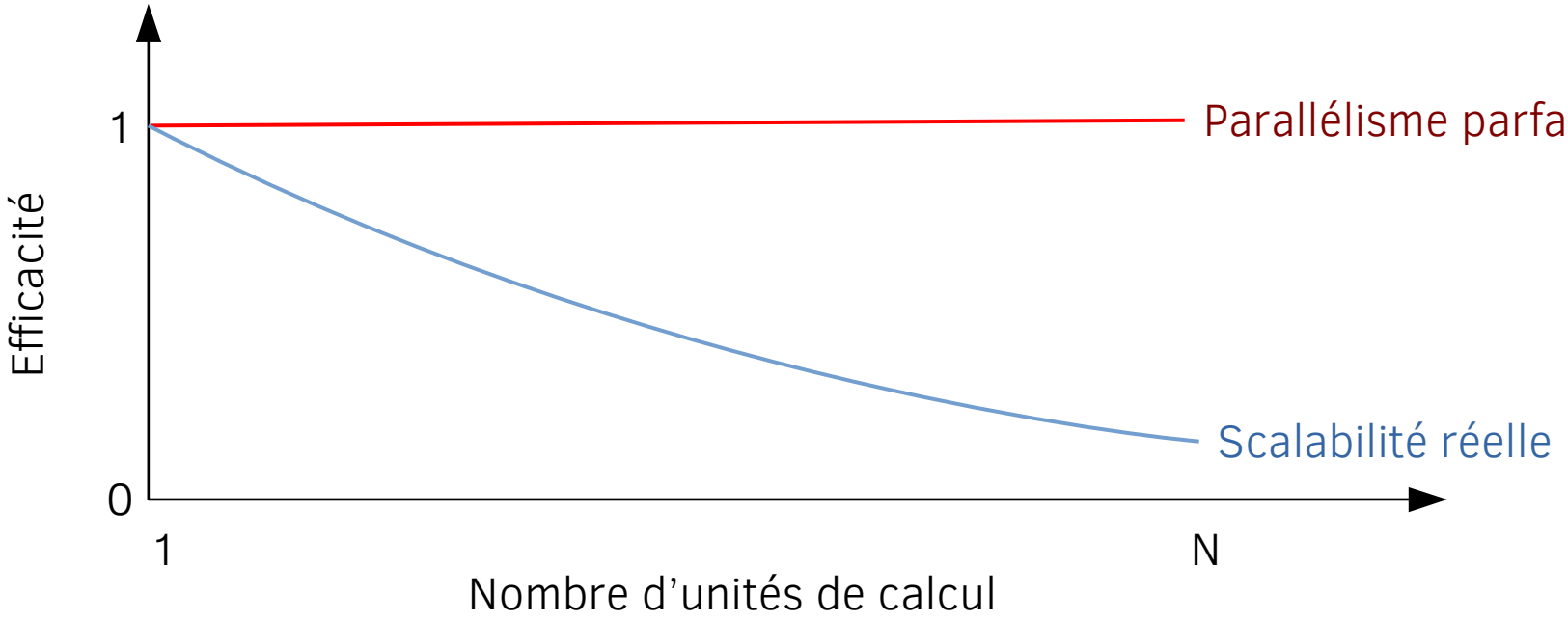
https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

Scalabilité ou passage à l'échelle forte

Strong Scalability

On trace généralement l'efficacité de scalabilité forte :
t(1) est le temps de simulation pou un processus
t(N) est le temps de simulation pour *N* processus

$$E_{forte}(N) = \frac{t(1)}{t(N) \times N}$$



https://www.sharcnet.ca/help/index.php/Measuring_Parallel_Scaling_Performance

Bien comprendre d'où vient la perte de scalabilité : loi d'Amdahl

La loi d'Amdahl : sur un seul processus, le temps de simulation peut être modélisé par la somme d'un temps passé hors des zones parallèles et un temps passé dans les régions parallèles.

- t(N) le temps de simulation pour N processus
- α_s la fraction du temps en dehors des régions parallèles
- α_p la fraction du temps dans les régions parallèles

$$t(1) = (\alpha_s + \alpha_p) t(1)$$

$$t(N) = (\alpha_s + \alpha_p / N) t(1)$$

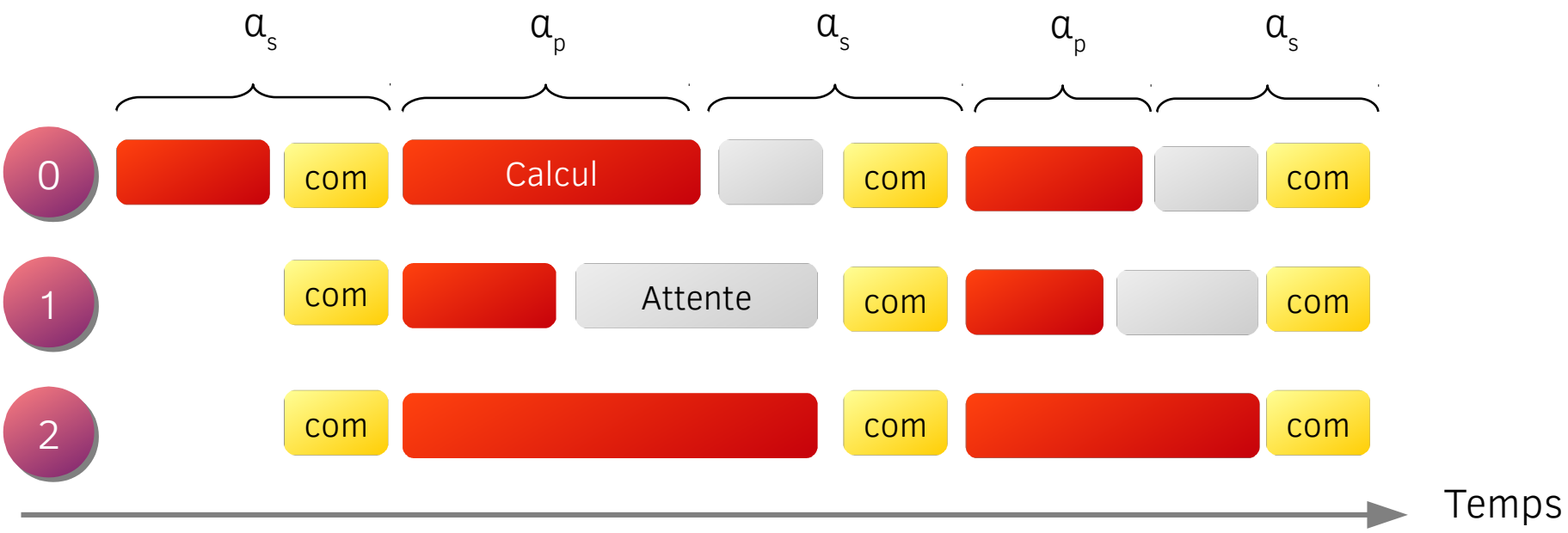
Bien comprendre d'où vient la perte de scalabilité : loi d'Amdahl

A partir de ce modèle, on obtient l'accélération (*speed-up*) *A* d'un programme parallèle en fonction de la fraction du temps passé dans les zones non-parallèles et la fraction du temps passé dans les zones parallèles pour un nombre donné de processus *N*.

$$A(N) = \frac{1}{\alpha_s + (1-\alpha_s)/N}$$

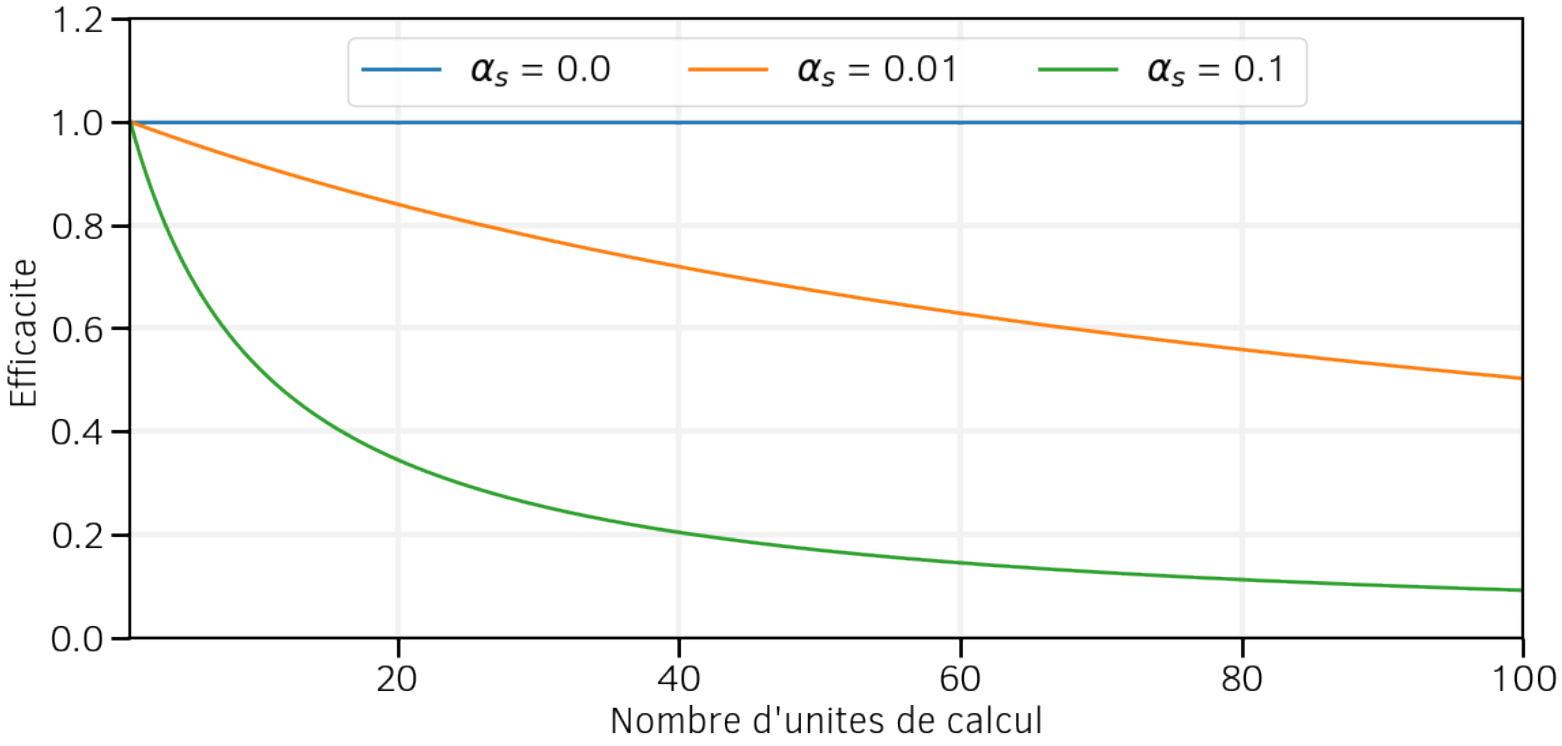
Bien comprendre d'où vient la perte de scalabilité : loi d'Amdahl

La part hors zone parallèle α_s représente le temps passé en séquentiel (seul un processus travaille), les temps de synchronisation, les temps de communication, le temps induit par les déséquilibres.



Bien comprendre d'où vient la perte de scalabilité : loi d'Amdahl

- Pour une fraction séquentielle donnée α_s, plus le nombre de processus est élevé, plus le temps séquentiel devient dominant et le speed-up apporté par la parallélisation limité
- Pour un nombre de processus donné *N*, plus la part séquentielle α_s est important, plus vite sera atteinte la limite de scalabilité d'un code parallèle



Bien comprendre d'où vient la perte de scalabilité : loi d'Amdahl

En fonction de la scalabilité d'un code parallèle, il arrive donc un moment où augmenter le nombre de processeur n'accélère plus l'exécution du code.

Mesure du temps : MPI_WTIME

- `MPI_WTIME` permet de récupérer le temps écoulé sur le processus courant en seconde


.f90

```
Real :: time
Time = MPI_WTIME()
```

- Par deux appels et une soustraction, cette fonction permet de déterminer le temps passer dans une section du code


.f90

```
Real :: time
time = MPI_WTIME()

! Des calculs...
…

! Ce temps est le temps passé entre les deux appels à MPI_WTIME
time = MPI_WTIME() - time
```

https://www.open-mpi.org/doc/v4.0/man3/MPI_Wtime.3.php