Cours 2.5 : MPI suite

Charles Bouillaguet
(charles.bouillaguet@univ-lille.fr)

2021-02-12

Durée des communications ?

Envoi d'un message du rang *i* au rang *j*

$$T = \alpha + n \times \beta$$

- $n =$ taille des données (bit)
- $\alpha =$ latence (s)
- $\beta \approx$ débit (s / bit).

$1/\beta$: bit / s (bande passante).

Modèle **simplifié** et **optimiste** !

- T indépendant de i et j (topologie ?)
- Indépendant des autres envois (congestion ?)

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs
taurus	lyon	2012	10Gbit ethernet	1156Mo/s	25.9μs

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs
taurus	lyon	2012	10Gbit ethernet	1156Mo/s	25.9μs
gros	nancy	2019	25Gbit ethernet	2480Mo/s	8.9μs

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs
taurus	lyon	2012	10Gbit ethernet	1156Mo/s	25.9μs
gros	nancy	2019	25Gbit ethernet	2480Mo/s	8.9μs
grcinq	nancy	2013	56Gbit InfiniBand	2488Mo/s	1.2μs

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs
taurus	lyon	2012	10Gbit ethernet	1156Mo/s	25.9μs
gros	nancy	2019	25Gbit ethernet	2480Mo/s	8.9μs
grcinq	nancy	2013	56Gbit InfiniBand	2488Mo/s	1.2μs
grimoire	nancy	2016	56Gbit InfiniBand	4248Mo/s	1.1μs

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs
taurus	lyon	2012	10Gbit ethernet	1156Mo/s	25.9μs
gros	nancy	2019	25Gbit ethernet	2480Mo/s	8.9μs
grcinq	nancy	2013	56Gbit InfiniBand	2488Mo/s	1.2μs
grimoire	nancy	2016	56Gbit InfiniBand	4248Mo/s	1.1μs
drac	grenoble	2015	100Gbit InfiniBand	7760Mo/s	1.7μs

MPI : simple et efficace

```
int MPI_Send(const void* buf, int count, MPI_Datatype datatype,
             int dest, int tag, MPI_Comm comm);

int MPI_Recv(void* buf, int count, MPI_Datatype datatype,
             int source, int tag, MPI_Comm comm,
             MPI_Status *status);
```

Cluster	Site	Année	Interface	Bandwith	Latency
PPTI	jussieu	20??	1Gbit ethernet	64Mo/s	50.0μs
sagitaire	lyon	2006	1Gbit ethernet	116Mo/s	65.0μs
taurus	lyon	2012	10Gbit ethernet	1156Mo/s	25.9μs
gros	nancy	2019	25Gbit ethernet	2480Mo/s	8.9μs
grcinq	nancy	2013	56Gbit InfiniBand	2488Mo/s	1.2μs
grimoire	nancy	2016	56Gbit InfiniBand	4248Mo/s	1.1μs
drac	grenoble	2015	100Gbit InfiniBand	7760Mo/s	1.7μs
grvingt	nancy	2018	100Gbit OmniPath	12100Mo/s	0.9μs

(mesure : ping-pong entre deux noeuds)

MPI : fonctions utiles

```
double MPI_Wtime();
```

Secondes écoulées depuis un point quelconque du passé, avec beaucoup de chiffres significatifs (mesure précise).

MPI : échange de message

Solution naïve (`MPI_Send` puis `MPI_Send`) $\rightsquigarrow$ interblocage

```
int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
                 int dest, int sendtag,
                 void *recvbuf, int recvcount, MPI_Datatype recvtype,
                 int source, int recvtag,
                 MPI_Comm comm, MPI_Status *status);
```

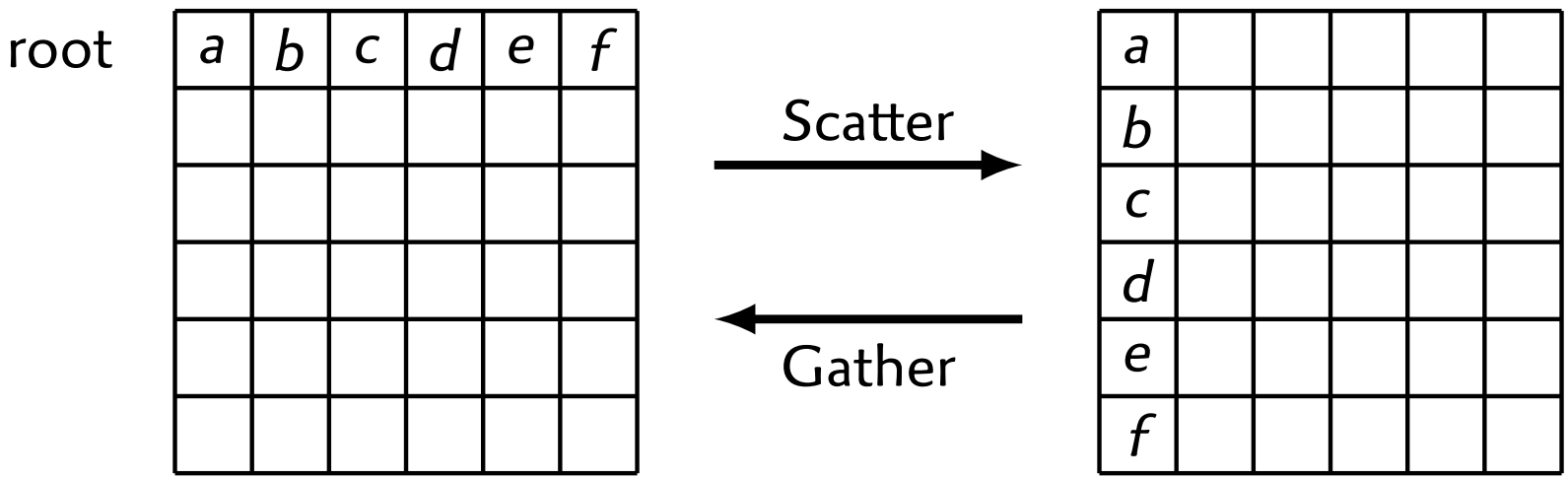
Les deux buffer doivent être distincts et ne pas se chevaucher.

```
int MPI_Sendrecv_replace(void* buf, int count, MPI_Datatype datatype,
                         int dest, int sendtag, int source, int recvtag,
                         MPI_Comm comm, MPI_Status *status);
```

Risque d'allouer de la mémoire

Gather / Scatter

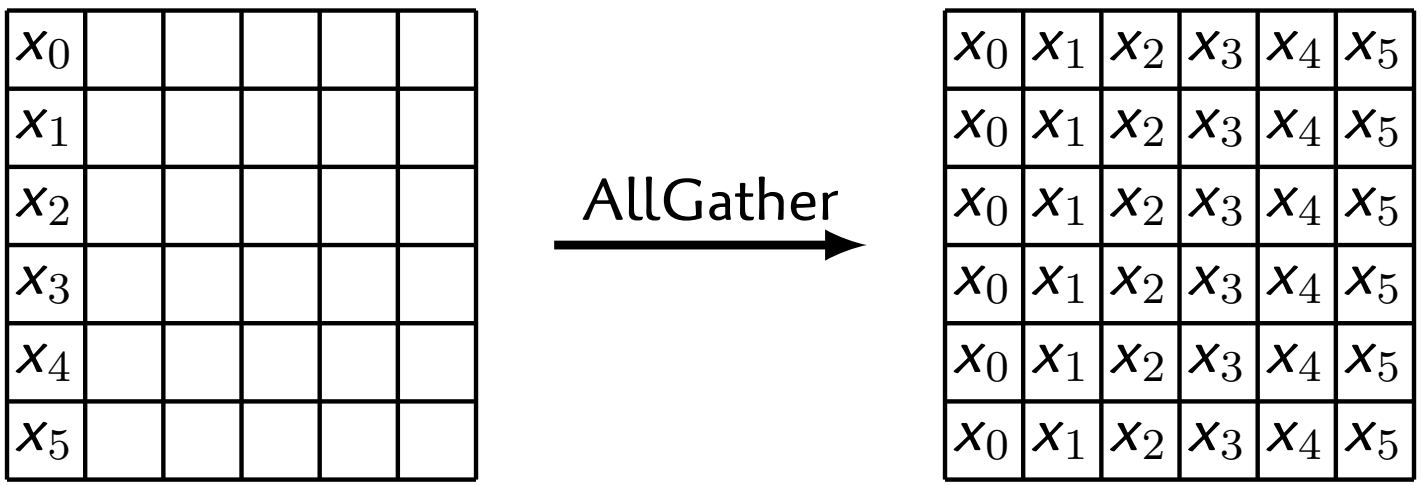
```
int MPI_Gather(const void* sendbuf, int sendcount, MPI_Datatype sendtyp
               void* recvbuf, int recvcount, MPI_Datatype recvtype,
               int root, MPI_Comm comm);
int MPI_Scatter(const void* sendbuf, int sendcount, MPI_Datatype sendty
                void* recvbuf, int recvcount, MPI_Datatype recvtype,
                int root, MPI_Comm comm);
```



- See also : `MPI_Gatherv`, `MPI_Scatterv`

MPI : AllGather

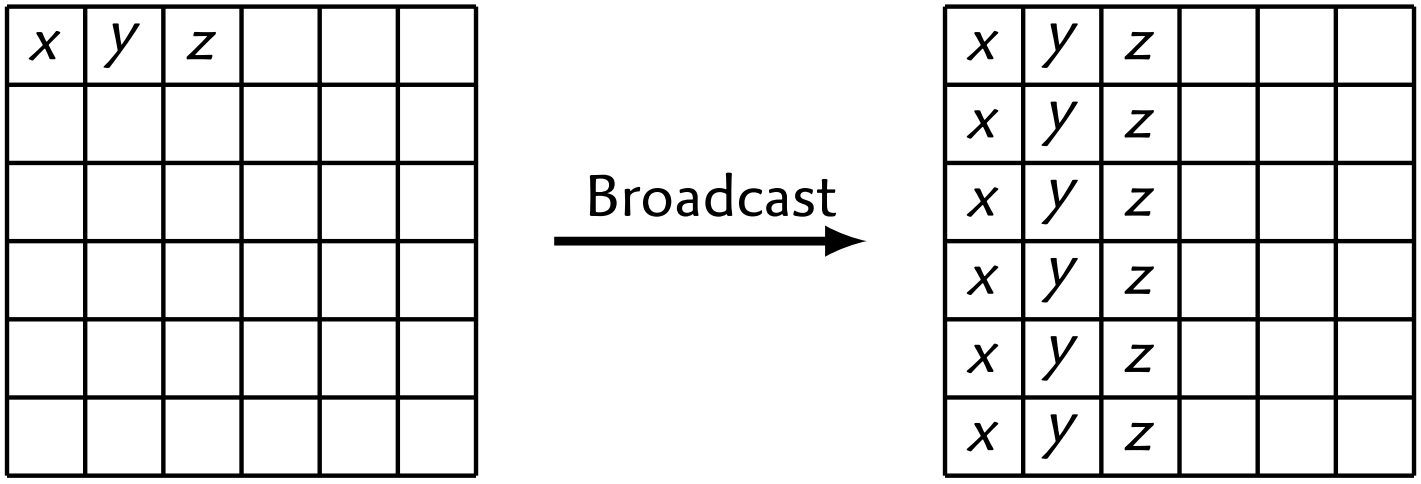
```
int MPI_Allgather(const void* sendbuf, int sendcount, MPI_Datatype send
                  void* recvbuf, int recvcount, MPI_Datatype recvtype,
                  MPI_Comm comm);
```



- See also : `MPI_Allgatherv`

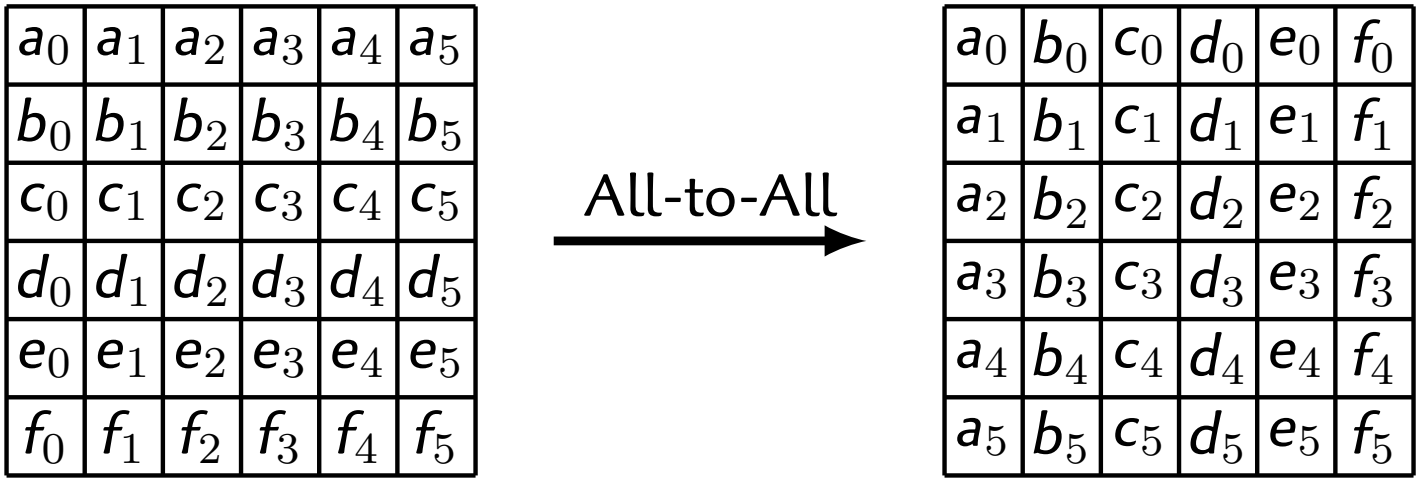
MPI : Broadcast

```
int MPI_Bcast(void* buffer, int count, MPI_Datatype datatype,
              int root, MPI_Comm comm);
```



MPI : All-to-All

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```



- See also : `MPI_Alltoallv`, `MPI_Alltoallw`

MPI : routines de communication collective

Usage typique

Broadcast	Diffusion de données communes avant les calculs
Scatter	Dispersion de données au début du calcul
Gather	Rassemblement des données à la fin du calcul
AllGather	Synchronisation entre deux phases de calcul
All-to-All	idem

MPI : faite souffrir votre switch

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- 16 noeuds, 4Go par noeud, 64Go en tout
- Chaque noeud envoie 256Mo à chaque autre

Cluster	Site	Année	Interface	T	aggregated BW
PPTI	jussieu	20??	1Gbit ethernet	254s	250Mo/s

MPI : faite souffrir votre switch

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- 16 noeuds, 4Go par noeud, 64Go en tout
- Chaque noeud envoie 256Mo à chaque autre

Cluster	Site	Année	Interface	T	aggregated BW
PPTI	jussieu	20??	1Gbit ethernet	254s	250Mo/s
sagitaire	lyon	2006	1Gbit ethernet	?	

MPI : faite souffrir votre switch

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- 16 noeuds, 4Go par noeud, 64Go en tout
- Chaque noeud envoie 256Mo à chaque autre

Cluster	Site	Année	Interface	T	aggregated BW
PPTI	jussieu	20??	1Gbit ethernet	254s	250Mo/s
sagitaire	lyon	2006	1Gbit ethernet	?	
paravance	rennes	2015	10Gbit ethernet	14.9s	4Go/s

MPI : faite souffrir votre switch

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- 16 noeuds, 4Go par noeud, 64Go en tout
- Chaque noeud envoie 256Mo à chaque autre

Cluster	Site	Année	Interface	T	aggregated BW
PPTI	jussieu	20??	1Gbit ethernet	254s	250Mo/s
sagitaire	lyon	2006	1Gbit ethernet	?	
paravance	rennes	2015	10Gbit ethernet	14.9s	4Go/s
gros	nancy	2019	25Gbit ethernet	6.8s	9.4Go/s

MPI : faite souffrir votre switch

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- 16 noeuds, 4Go par noeud, 64Go en tout
- Chaque noeud envoie 256Mo à chaque autre

Cluster	Site	Année	Interface	T	aggregated BW
PPTI	jussieu	20??	1Gbit ethernet	254s	250Mo/s
sagitaire	lyon	2006	1Gbit ethernet	?	
paravance	rennes	2015	10Gbit ethernet	14.9s	4Go/s
gros	nancy	2019	25Gbit ethernet	6.8s	9.4Go/s
grcinq	nancy	2013	56Gbit InfiniBand	4.8s	13Go/s

MPI : faite souffrir votre switch

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- 16 noeuds, 4Go par noeud, 64Go en tout
- Chaque noeud envoie 256Mo à chaque autre

Cluster	Site	Année	Interface	T	aggregated BW
PPTI	jussieu	20??	1Gbit ethernet	254s	250Mo/s
sagitaire	lyon	2006	1Gbit ethernet	?	
paravance	rennes	2015	10Gbit ethernet	14.9s	4Go/s
gros	nancy	2019	25Gbit ethernet	6.8s	9.4Go/s
grcinq	nancy	2013	56Gbit InfiniBand	4.8s	13Go/s
grvingt	nancy	2018	100Gbit OmniPath	2.2s	29Go/s

MPI : avec le *vrai* matériel de HPC

```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- n noeuds, 6.75Go par noeud, $6.75n$Go en tout
- Chaque noeud envoie 256Mo à chaque autre

MPI : avec le *vrai* matériel de HPC

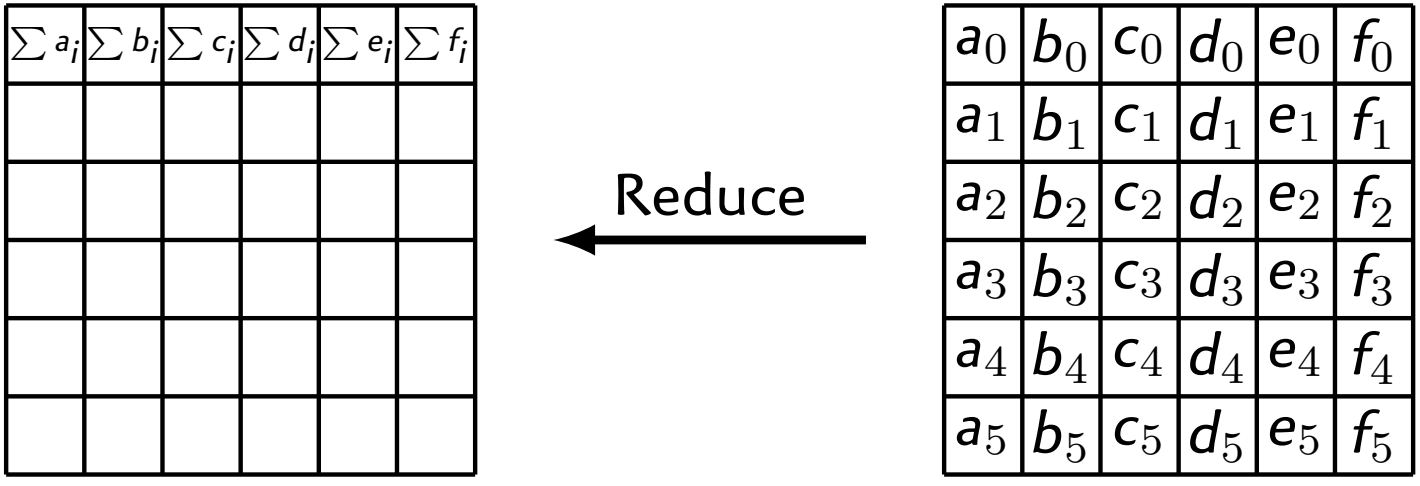
```
int MPI_Alltoall(const void* sendbuf, int sendcount, MPI_Datatype sendtype,
                 void* recvbuf, int recvcount, MPI_Datatype recvtype,
                 MPI_Comm comm);
```

- n noeuds, 6.75Go par noeud, $6.75n$Go en tout
- Chaque noeud envoie 256Mo à chaque autre

# nodes	Data	T (s)	Aggregated BW
2	13.5Go	2.7	6.8Go/s
4	27Go	2.9	9.3Go/s
⋮	⋮	⋮	⋮
64	430Go	4.1	105Go/s
128	861Go	8.9	96Go/s
256	1.7To	13.0	130Go/s
512	3.4To	13.7	248Go/s
1024	6.9To	15.9	434Go/s
2048	13.4To	17.0	788Go/s
4096	27.5To	18.7	1470Go/s

MPI : Reduce

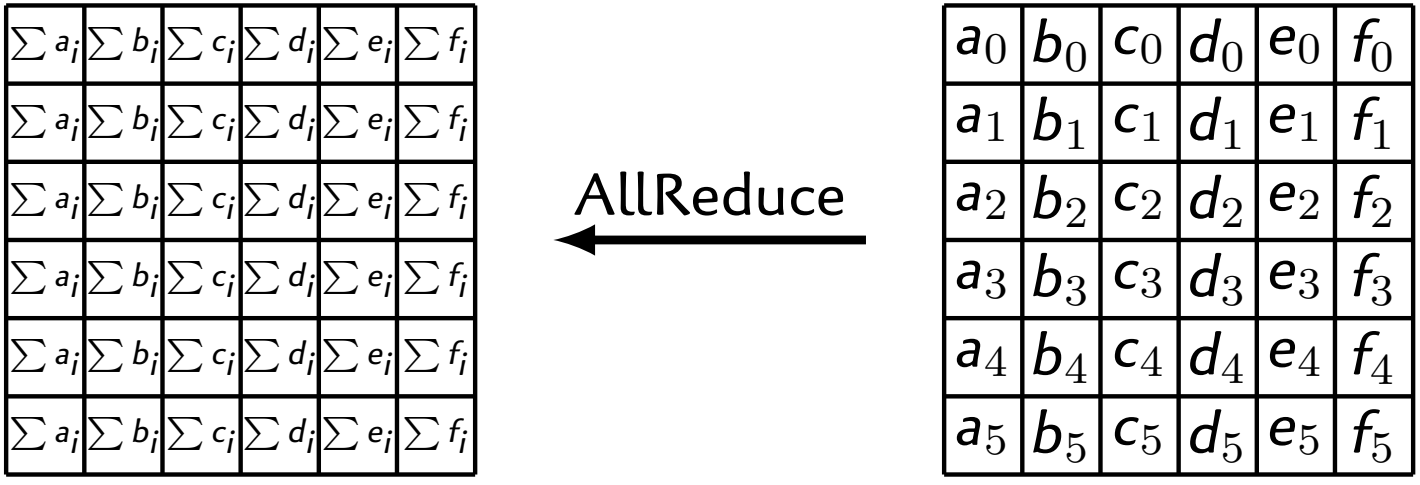
```
int MPI_Reduce(const void *sendbuf, void *recvbuf, int count,
               MPI_Datatype datatype, MPI_Op op, int root,
               MPI_Comm comm);
```



- Opérations prédéfinies : $+$, $\times$, $\min$, $\max$, et, ou, non, ...
- Opérations définissables par l'utilisateur

MPI : AllReduce

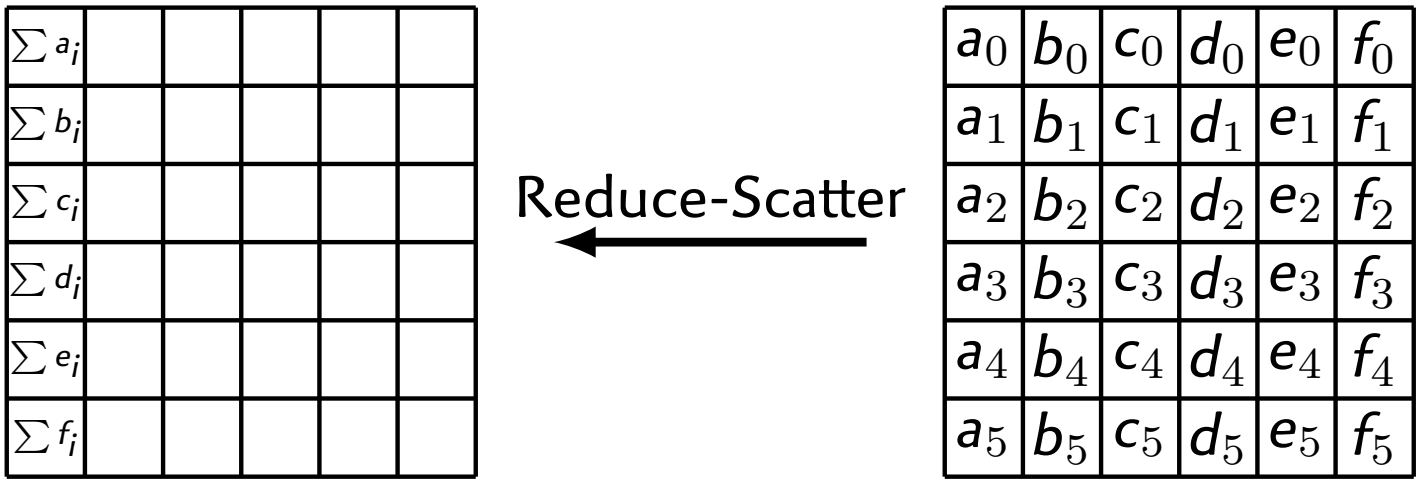
```
int MPI_Allreduce(const void *sendbuf, void *recvbuf, int count,
                  MPI_Datatype datatype, MPI_Op op, MPI_Comm comm);
```



- Opérations prédéfinies : $+$, $\times$, $\min$, $\max$, et, ou, non, ...
- Opérations définissables par l'utilisateur

MPI : Reduce-Scatter

```
int MPI_Reduce_scatter_block(const void* sendbuf, void* recvbuf,
                             int recvcount, MPI_Datatype datatype,
                             MPI_Op op, MPI_Comm comm)
```



- Opérations prédéfinies : $+$, $\times$, $\min$, $\max$, et, ou, non, ...
- Opérations définissables par l'utilisateur
- See also : `MPI_Reduce_scatter`

MPI : Scan

A.k.a. Prefix-sum

```
int MPI_Scan(const void* sendbuf, void* recvbuf, int count,
             MPI_Datatype datatype, MPI_Op op, MPI_Comm comm);
```

a_0	b_0	c_0
a_0+a_1	b_0+b_1	c_0+c_1
$a_0+a_1+a_2$	$b_0+b_1+b_2$	$c_0+c_1+c_2$
$a_0+\cdots+a_3$	$b_0+\cdots+b_3$	$c_0+\cdots+c_3$
$a_0+\cdots+a_4$	$b_0+\cdots+b_4$	$c_0+\cdots+c_4$
$a_0+\cdots+a_5$	$b_0+\cdots+b_5$	$c_0+\cdots+c_5$

Scan ←

a_0	b_0	c_0
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4
a_5	b_5	c_5

- Opérations prédéfinies : $+$, $\times$, $\min$, $\max$, et, ou, non, ...
- Opérations définissables par l'utilisateur
- See also : `MPI_Exscan`

MPI_IN_PLACE

- Certaines opérations collectives doivent copier des données du buffer de départ vers le buffer de destination.
- Au moins sur `root`, parfois chez tout le monde.

Pénible ?

- `MPI_IN_PLACE` à la place du buffer d'envoi désactive la copie.
- Les données sont *lues* (et écrites) dans le buffer de destination, là où elles auraient dû arriver.
- Fonctionne dans Scatter, Gather, Allgather, All-to-all, reduce, scan, etc.

Un aspect pratique

- `MPI_Scatter` / `MPI_Gather` / `MPI_Alltoall` / ...
  - Même taille envoyée/reçue par tout le monde
  - Très pratique si $N \,\%\, p = 0$.

Si $N \,\%\, p \neq 0$?

- `MPI_Scatterv` / `MPI_Gatherv` / `MPI_Alltoallv` / ...
  - Tailles individualisées (mais encore plus d'arguments !)
- Solution de facilité : bourrage
  - Augmenter N jusqu'à $p\lceil N/p \rceil$ en rajoutant... des zéros ?
  - Chaque « tranche » de taille $\lceil N/p \rceil$.
  - La dernière contient une portion inutile.

Sujets non abordés

- Types personnalisés
- IO parallèle
- One-sided communications
  - Mais on verra peut-être OpenSHMEM
- Création/gestion des intra/inter-communicateurs
- Opérations collectives asynchrones
- ...

Sujets non abordés

- Types personnalisés
- IO parallèle
- One-sided communications
  - Mais on verra peut-être OpenSHMEM
- Création/gestion des intra/inter-communicateurs
- Opérations collectives asynchrones
- ...

RTFM