

Message Passing Interface

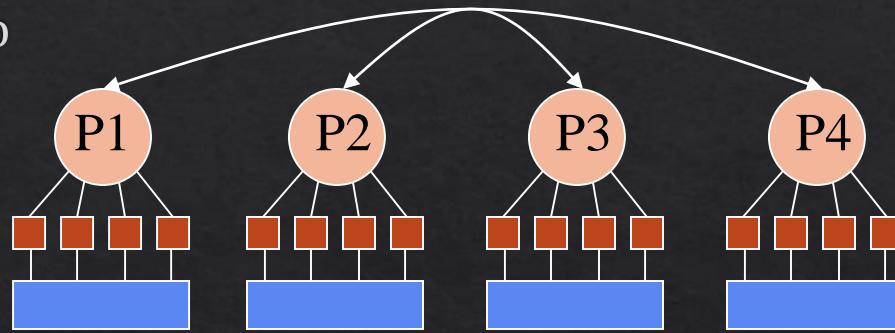
F.CS306 Parallel programming – Lecture 14

Хичээлийн агуулга

- ❖ Мессеж дамжуулалт (Message-passing) загвар
 - ❖ Мессеж дамжуулалт
 - ❖ Мэдээлэл солилцооны хэлбэрүүд
- ❖ MPI мессеж дамжуулалт
 - ❖ Hello, World!
 - ❖ Онолын ойлголтууд
 - ❖ Өгөгдлийн төрөл, Tag
 - ❖ Send/Receive

Message-Passing загвар

- ❖ Процесс нь программын тоолуур ба хаягийн хүснэгт
- ❖ Процессууд дундын хаягийн хүснэгттэй олон *thread* (программын тоолуурууд болон харгалзах стек)-тэй байж болно



- ❖ MPI бол процесс (thread биш, тусдаа хаягийн хүснэгттэй) хоорондын мэдээлэл солилцоно
- ❖ Процесс хооронодын мэдээллэл солилцоо:
 - ❖ Синхрончлол
 - ❖ Нэг нөгөөгийнх рүү чиглэсэн өгөгдөл шилжүүлэлт

Параллел тооцооллын загварууд

- ❖ Өгөгдлийн паралелизм
 - ❖ Өгөгдлийн элементүүд дээр зэрэгцээ ажиллах ижил зааварууд (SIMD)
- ❖ Даалгаврын паралелизм
 - ❖ Өөр өөр өгөгдөл дээрх өөр өөр заавар (MIMD)
- ❖ SPMD (single program, multiple data)
 - ❖ Үйлдлийн түвшинд синхрончлогдоогүй
- ❖ MIMD программ бүр нь SPMD байж чадах учраас SPMD ба MIMD хоёрыг эквивалент гэж хэлж болно (similarly for SIMD, but not in practical sense)

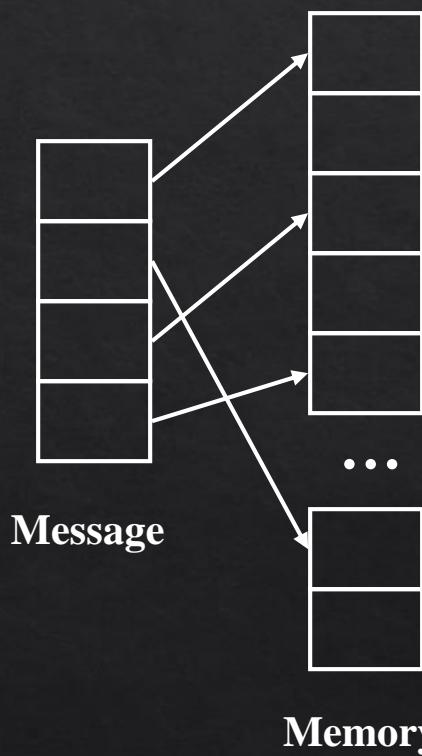
MPI нь MIMD/SPMD паралелизм.

Мессеж дамжуулалт

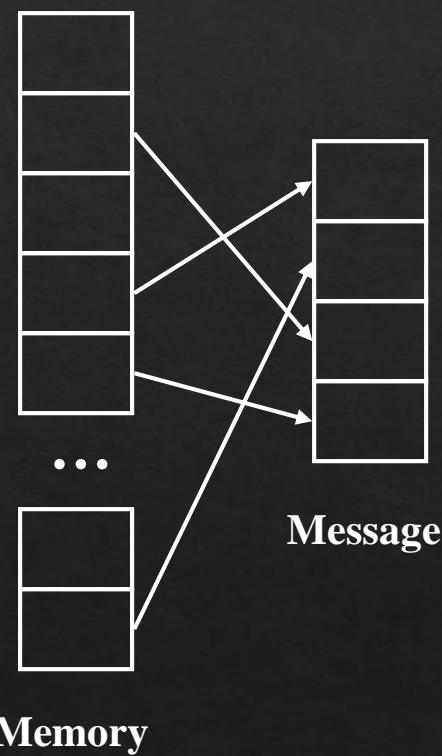
- ❖ Мессеж дамжуулалт/Message Passing:
 - ❖ *Send*: Захидал илгээхтэй төстэй
 - ❖ *Receive*: Шуудангийн хайрцгаас захидал авахтай адил
 - ❖ *Scatter-gather*: Мессеж дэх өгөгдлийн элементүүдийг санах ойн олон байршилд “тараах (scatter)”, өгөгдлийн элементүүдийг санах ойн олон байршилаас “цуглуулах (gather)”
- ❖ Network performance:
 - ❖ *Latency*: Send ажиллаж эхлхээс Receive эхний байтыг хүлээн авах хүртэлх хугацаа
 - ❖ *Bandwidth*: Илгээгчийн хүлээн авагч руу өгөгдөл дамжуулах чадал.

Scatter-Gather

Scatter (Receive)

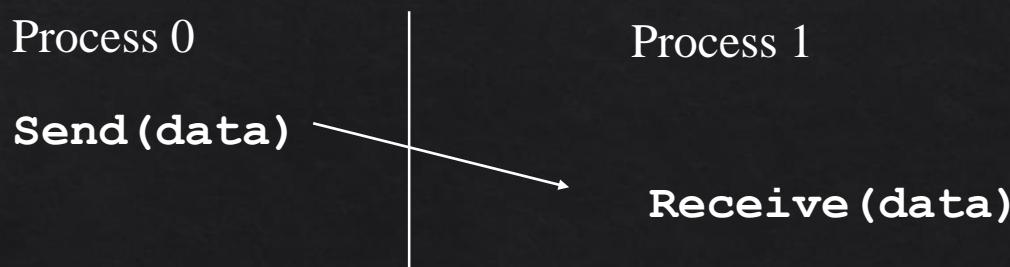


Gather (Send)



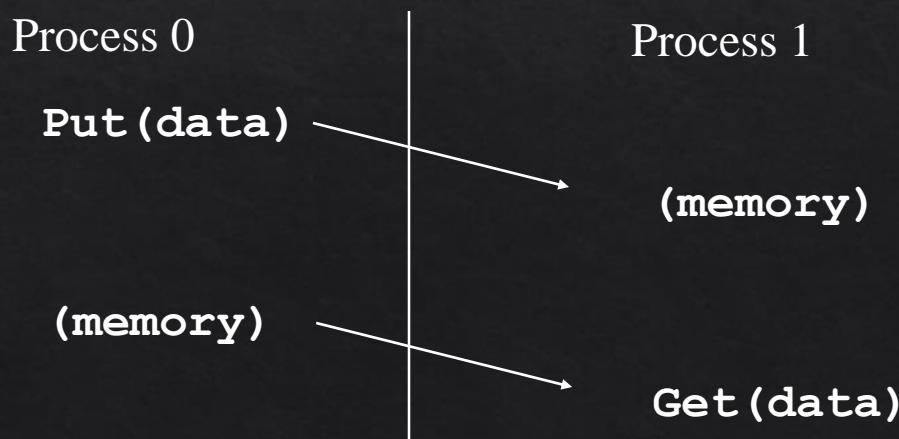
Мэдээлэл солилцооны Cooperative (хамтарсан) үйлдэл

- ❖ message-passing хандлага → өгөгдөл солилцоо - хамтын ажил
- ❖ Өгөгдлийг нэг процесс нь *илгээж*, нөгөө нь *хүлээн авдаг*.
- ❖ Давуу тал: хүлээн авагч процесийн санах ой дахь өөрчлөлтийг хүлээн зөвхөн авагчийн оролцоогоор л гүйцэтгэнэ.
- ❖ Мэдээлэл солилцоо болон Синхрончлол хоёр нь нэг багц
- Push model (active data transfer)



Мэдээлэл солилцооны One-Sided (нэг талт) үйлдэл

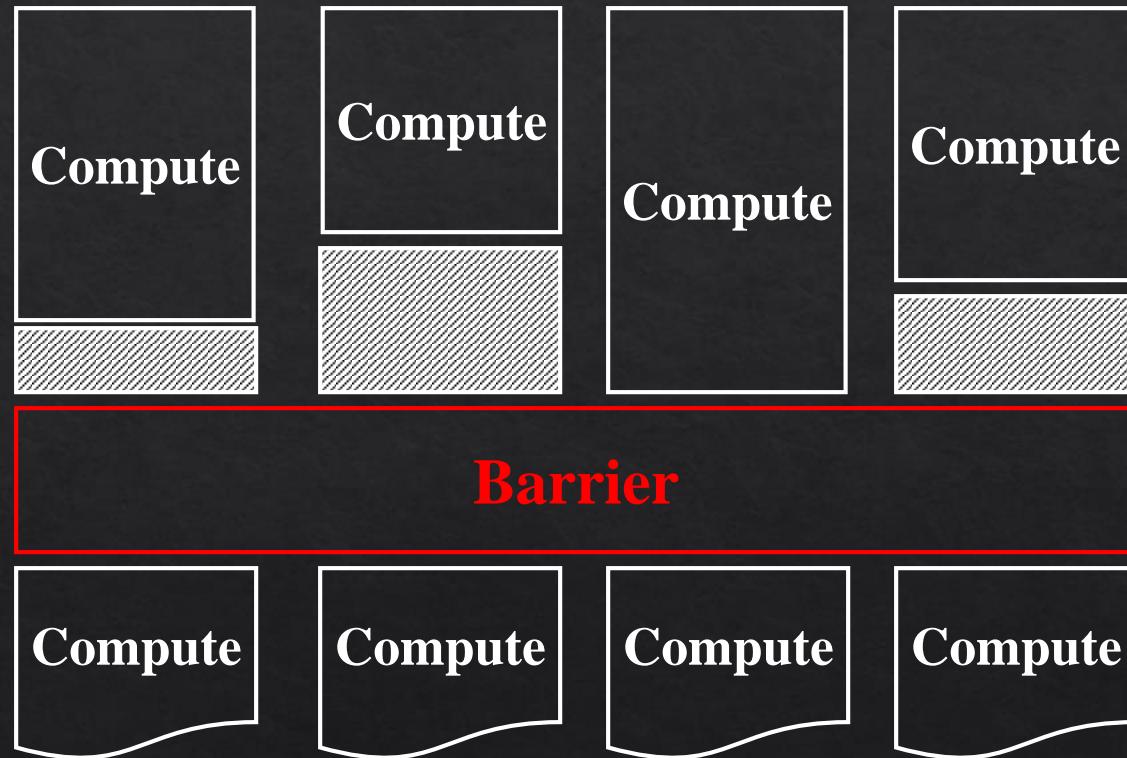
- ❖ One-sided нь процесс хоорондын үйлдэл
 - ❖ Алсын (remote) санах ой дээрх уншилт, бичилт
 - ❖ Зөвхөн нэг процесс л оролцоход хангалттай
 - ❖ Давуу тал: Мэдээлэл солилцоо ба синхрончлол хоёрыг салгах
 - ❖ MPI-2 дээр дэмжигддэг.
- Pull model (passive data transfer) for get



МЭДЭЭЛЭЛ СОЛИЛЦООНЫ ХЭЛБЭР

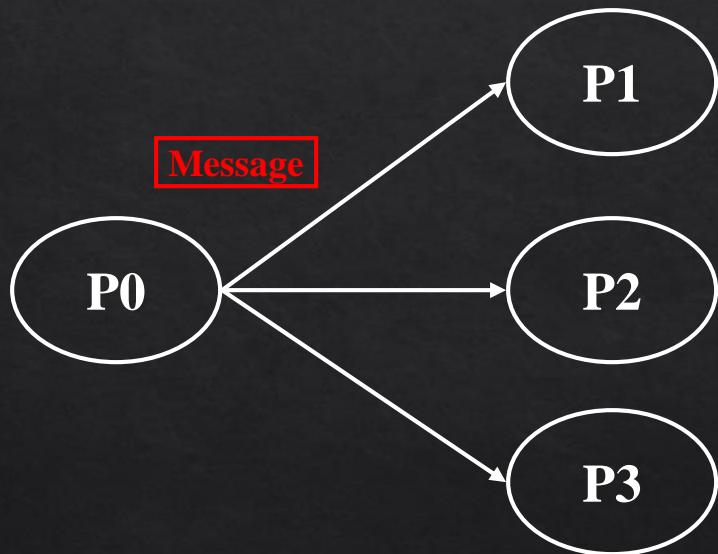
- ❖ Хос процесс хоорондох мэдээлэл солилцоо
- ❖ Олон процесс дундах *collective* мэдээлэл солилцоо
 - ❖ Процессын бүлэг: Хэд хэдэн процесс логикоор бүлэглэгдсэн
 - ❖ Бүлэг доторх мэдээлэл солилцоо
- ❖ *Collective үйлдлүүд:*
 - ❖ Barrier,
 - ❖ Broadcast (one-to-all),
 - ❖ multicast (one-to-many),
 - ❖ All-to-all,
 - ❖ Reduction (all-to-one)

Barrier

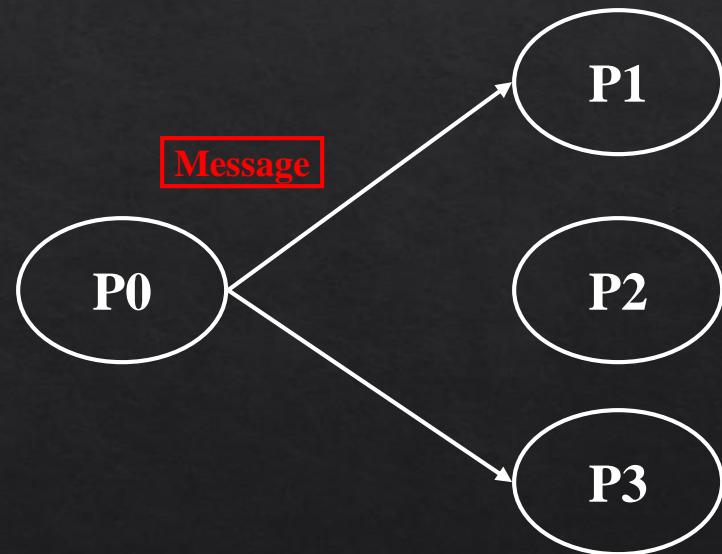


Broadcast and Multicast

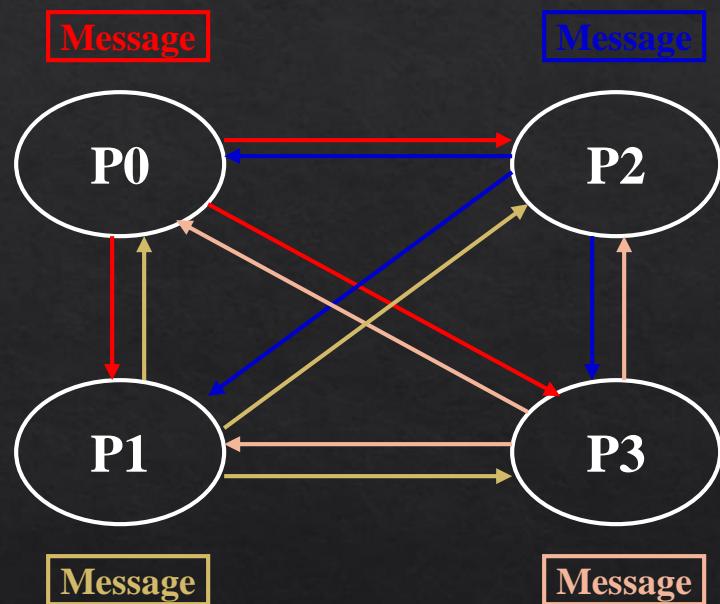
Broadcast



Multicast

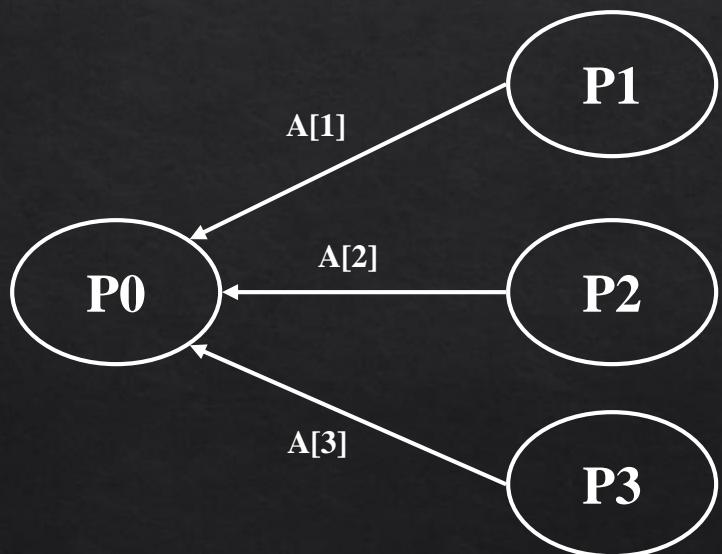


All-to-All

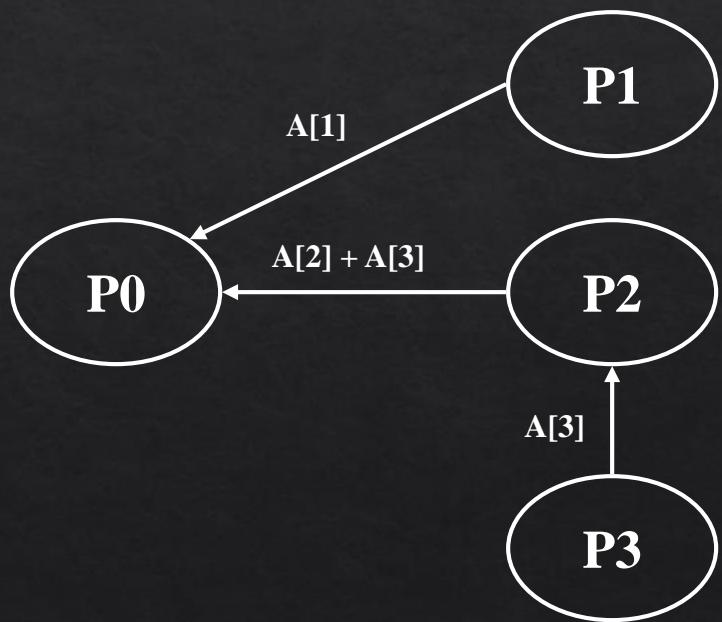


Reduction

```
sum ← 0  
for i ← 1 to p do  
    sum ← sum + A[i]
```



$$\begin{aligned} A[0] + A[1] + A[2] + A[3] &\leftarrow A[0] \\ &\leftarrow A[1] \\ &\leftarrow A[2] \\ &\leftarrow A[3] \end{aligned}$$



ЖИШЭЭ

```
#include "mpi.h"
#include <stdio.h>

int main(int argc, char *argv[])
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    printf( "I am %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

Зарим үндсэн ойлголт

- ❖ Процессуудыг *group*-үүдэд дотор багцалж болно.
- ❖ Мессеж бүрийг *context* дотор илгээнэ
 - ❖ Тухайн контексоор хүлээн авах ёстой
 - ❖ *Tag* ашигладаг
- ❖ *Group* ба *context* нь *communicator*-ыг бүрдүүлнэ.
 - ❖ *communicator* үүсгэх: **MPI_Comm_split**
- ❖ Процессийг *rank* ашиглан ялгаж танидаг.
 - ❖ Тухайн *Communicator*-т хамаарах *group*-д суурилж дугаарлагддаг.
- ❖ Default *communicator*: **MPI_COMM_WORLD**
 - ❖ Бүх анхдагч процессыг агуулдаг

MPI өгөгдлийн төрөл

- ❖ Мессеж өгөгдлийн дүрслэл:
 - ❖ Хаяг, хэмжээ, өгөгдлийн төрөл
- ❖ MPI *datatype*:
 - ❖ Программчлалын хэлэнд урьдчилан тодорхойлсон төрөл
 - ❖ MPI өгөгдлийн төрлийн шугаман матриц
 - ❖ Өгөгдлийн төрлүүдээр нэгтгэсэн блок
 - ❖ Өгөгдлийн төрлүүдээр үүсгэсэн бүтэцтэй блокын матриц
 - ❖ Өгөгдлийн төрлийн дурын бүтэц
- ❖ MPI нь өгөгдлийн бүтэц байгуулах функциудтэй
 - ❖ Array of (int, float) pairs
 - ❖ Row of a matrix stored columnwise

MPI Tag

- ❖ Мессежүүд нь хэрэглэгчийн тодорхойлсон *tag* гэсэн бүхэл тоотой хамт илгээгддэг
 - ❖ Зурвасыг хүлээн авахад таних зорилгоор
- ❖ Мессежийн төгсгөлд тусгай *tag* тодорхойлж болно:
 - ❖ **MPI_ANY_TAG** нь ямар tag-ийг зөвшөөрч хүлээн авна
- ❖ Ерөнхийдөө өгөгдлийн төрөлтэй андуурагдах учраас tag гэж нэрлэсэн

MPI Basic (Blocking) Send

MPI_SEND (start, count, datatype, dest, tag, comm)

- ❖ message буфер: **start**, **count**, **datatype**.
- ❖ Хандаж буй процесс: **dest**
 - ❖ Хандаж буй процесийн *rank*-ийн *communicator*: **comm**
- ❖ Функц ажиллаж дууссаны дараа
 - ❖ Өгөгдөл систем рүү илгээгдсэн байна
 - ❖ Буферийг дахин ажшиглах боломжтой болно

MPI Basic (Blocking) Receive

`MPI_RECV(start, count, datatype, source, tag, comm, status)`

- ❖ Системээс тохирох мессеж иртэл хүлээнэ.
 - ❖ **source** болон **tag**
 - ❖ Буфер чөлөөтэй байх ёстой
- ❖ **source** бол **comm** *communicator* дээрх *rank* байна
 - ❖ Эсвэл **MPI_ANY_SOURCE**
- ❖ **status** нь нэмэлт мэдээлэл
- ❖ **count** –аа бага урттайг хүлээж авна, илүүг нь хүлээж авахгүй.

MPI is Simple

- ❖ Many parallel programs can be written using just these six functions, only two of which are non-trivial:
 - ◊ **`MPI_INIT`**
 - ◊ **`MPI_FINALIZE`**
 - ◊ **`MPI_COMM_SIZE`**
 - ◊ **`MPI_COMM_RANK`**
 - ◊ **`MPI_SEND`**
 - ◊ **`MPI_RECV`**
- ❖ Point-to-point (send/recv) isn't the only way...

Thanks.