

Параллелчлалын ҮНДЭС

F.CS306 ПАРАЛЛЕЛ ПРОГРАММЧЛАЛ – Лекц 2

Г.ГАНБАТ
ganbatg@must.edu.mn

Хичээлийн агуулга

- ❖ Speedup, Efficiency, Scalability, болон Compute-to-Computation Ratio
- ❖ Тоог нэмэх параллел алгоритм, түүний дүн шинжилгээ
- ❖ *shared-memory* болон *distributed-memory* архитектур
- ❖ Параллел програм зохиоход анхаарах чухал зүйлс:
 - ❖ Partitioning, Communication, Synchronization, Load Balancing

Параллел алгоритмын шинжилгээ

Програм, алгоритмын ажиллагааг ойлгохын тулд шинжилгээ хийдэг

- ❖ Процессорын тоотой уялдуулан дараах шинжилгээний багцуудыг хэрэгжүүлдэг
 - ❖ Speedup
 - ❖ Efficiency and cost
 - ❖ Scalability
 - ❖ Computation-to-communication ratio

Хурдсалт (Speedup)

- ❖ Дараалсан программчлалаас хир хурдан ажиллаж байгааг шалгадаг
- ❖ Хурдсалтыг параллел код бүрийн эсвэл алгоритмын хувьд ерөнхийд нь тооцоолно.
- ❖ Хурдсалт (S) нь p ширхэг процессор дээр ажиллах хугацаа ($T(p)$) болон нэг процессор дээр ажиллах хугацаа ($T(1)$) – уудын харьцаа юм.

$$S = \frac{T(1)}{T(p)}$$

Үр ашиг ба өртөг (Efficiency and cost)

- ❖ Боломжит хамгийн өндөр хурдсалт - *шугаман хурдсалт*.
 - ❖ p ширхэг процессор эсвэл цөмийн тусламжтайгаар хүрч болох хамгийн дээд хурдсалт нь p .
- ❖ Хурдсалтыг процессорууд болон цөмүүдийн тоотой уялдуулах
- ❖ S -ийг p -д хувааж үр ашиг (Efficiency) E -г олно.
 - ❖ шугаман хурдсалт нь 100% ойролцоо утгатай болно

$$E = \frac{S}{p} = \frac{T(1)}{T(p) \times p}$$

- ❖ Өртөг (Cost) нь төстэй боловч ажиллах хугацаа $T(p)$ (хурдсалтын оронд) –г цөмийн тоо p -ээр үржүүлдэг.

$$C = T(p) \times p$$

Чадамж (Scalability)

- ❖ Ихэнхдээ тодорхой тооны процессор эсвэл цөмүүдийн үр ашгийг хэмжихийн оронд өөр өөр тооны хувьд шинждэг.
 - ❖ $P = 1, 2, 4, 8, 16, 32, 64, 128$
- ❖ *Strong scalability*: Процессоруудын тоог нэмэгдүүлж шинжлэх
- ❖ *Weak scalability*: Процессоруудын тоо болон оролтын өгөгдлийн хэмжээг өсгөж шинжилнэ

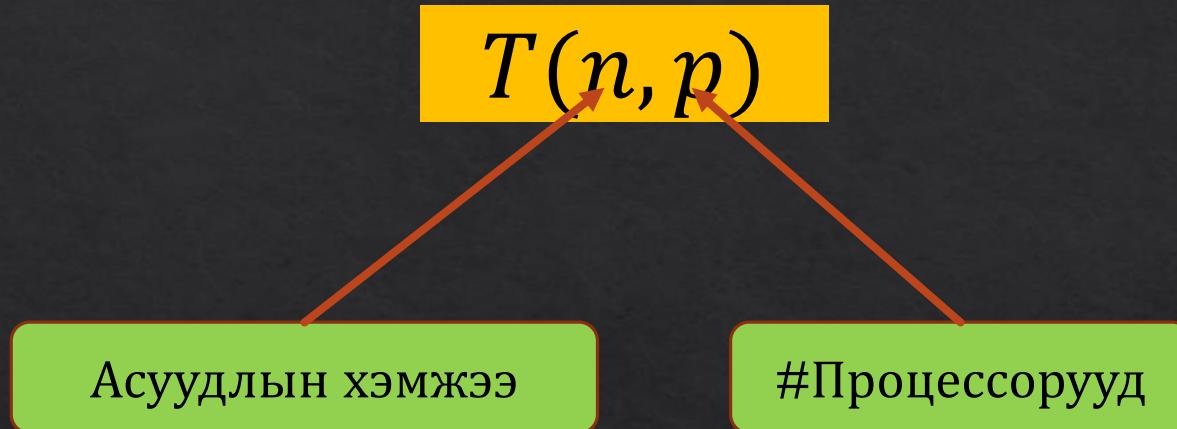
Computation-to-communication харьцаа

- ❖ Параллел хэрэгжүүлэлтийн боломжит чадамжид нөлөөлөх чухал хэмжигдэхүүн юм.
- ❖ Тооцоололд зарцуулсан хугацааг процессоруудын хооронд мэдээлэл солилцоооны хугацаагаар хувааж тооцоолно.
- ❖ Өндөр байх тусам хурдсалт, үр ашиг нэмэгдэнэ.

Тоог нэмэх параллел алгоритм

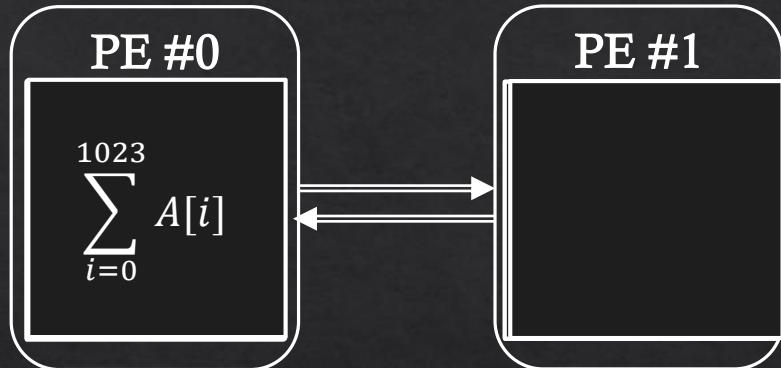
- ❖ **Оролт:** n ширхэг тоог агуулсан A массив
- ❖ **Гаралт:** $\sum_{i=0}^{n-1} A[i]$
- ❖ **Даалгавар:** Процессийн элементүүдийн (*processing elements – PEs*) массивийг ашиглан энэ асуудлыг үр ашигтай параллелчлах
- ❖ **Төсөөлөл:**
 1. **Тооцоолол:** РЕ бүр 1 сек-д санах ойд байх хоёр тоог нэмнэ
 2. **Мэдээлэл солилцоо:** РЕ нь санах ойгоосоо өөр РЕ-ийн санах ойд 3 сек-д өгөгдөл илгээнэ (хэмжээ хамаарахгүй)
 3. **Оролт ба гаралт:** Програм эхлээд оролтын массивийг РЕ #0 дээр байрлуулна. Гаралтыг мөн байрлалд цугуулна.
 4. **Синхрочлол:** Бүх РЕ нь lock-step зарчмаар ажилладаг; жны: Тэд compute, communicate эсвэл idle төлвийн нэгэнд байх ёстой.

Тоог нэмэх параллел алгоритм



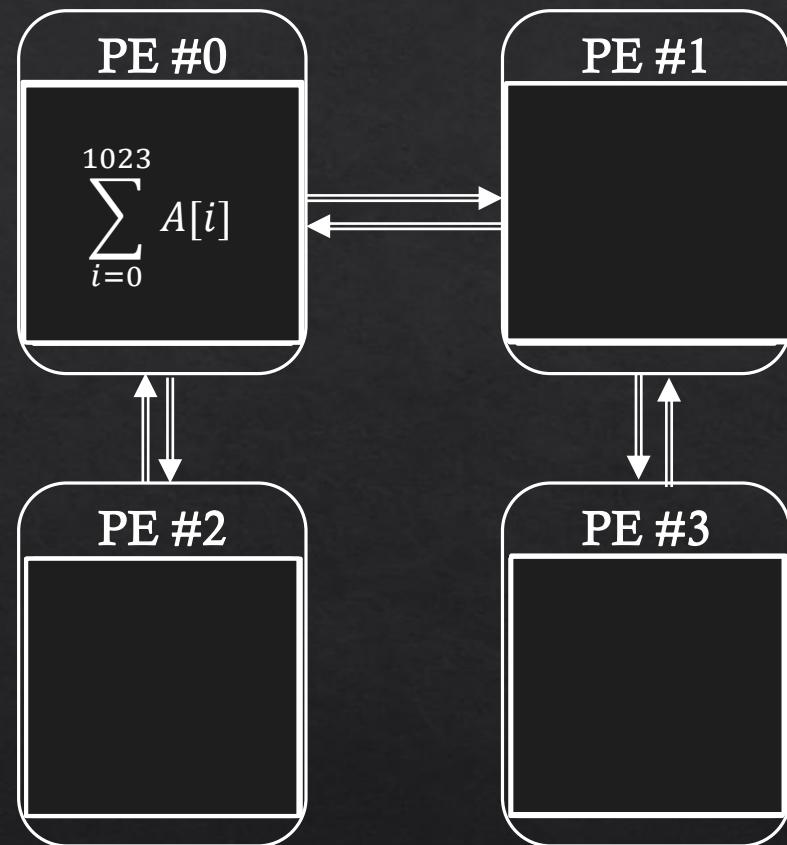
- ❖ Эхлээд дараалж ажиллахаар авна. ($p = 1$):
 - ❖ Манай жишээний хувьд: $T(1, n) = n - 1$ сек
- ❖ Бид юу сонирхож байна вэ?
 - ❖ Speedup: Бид $p > 1$ процессороор хэр хурдан вэ?
 - ❖ Efficiency: Бидний параллел программ үр ашигтай байна уу?
 - ❖ Scalability: Бидний параллел програм нь янз бүрийн тооны процессоруудад хэрхэн ажиллах вэ (тогтмол / янз бүрийн хэмжээтэй асуудлын хувьд)?

Тоог нэмэх параллел алгоритм



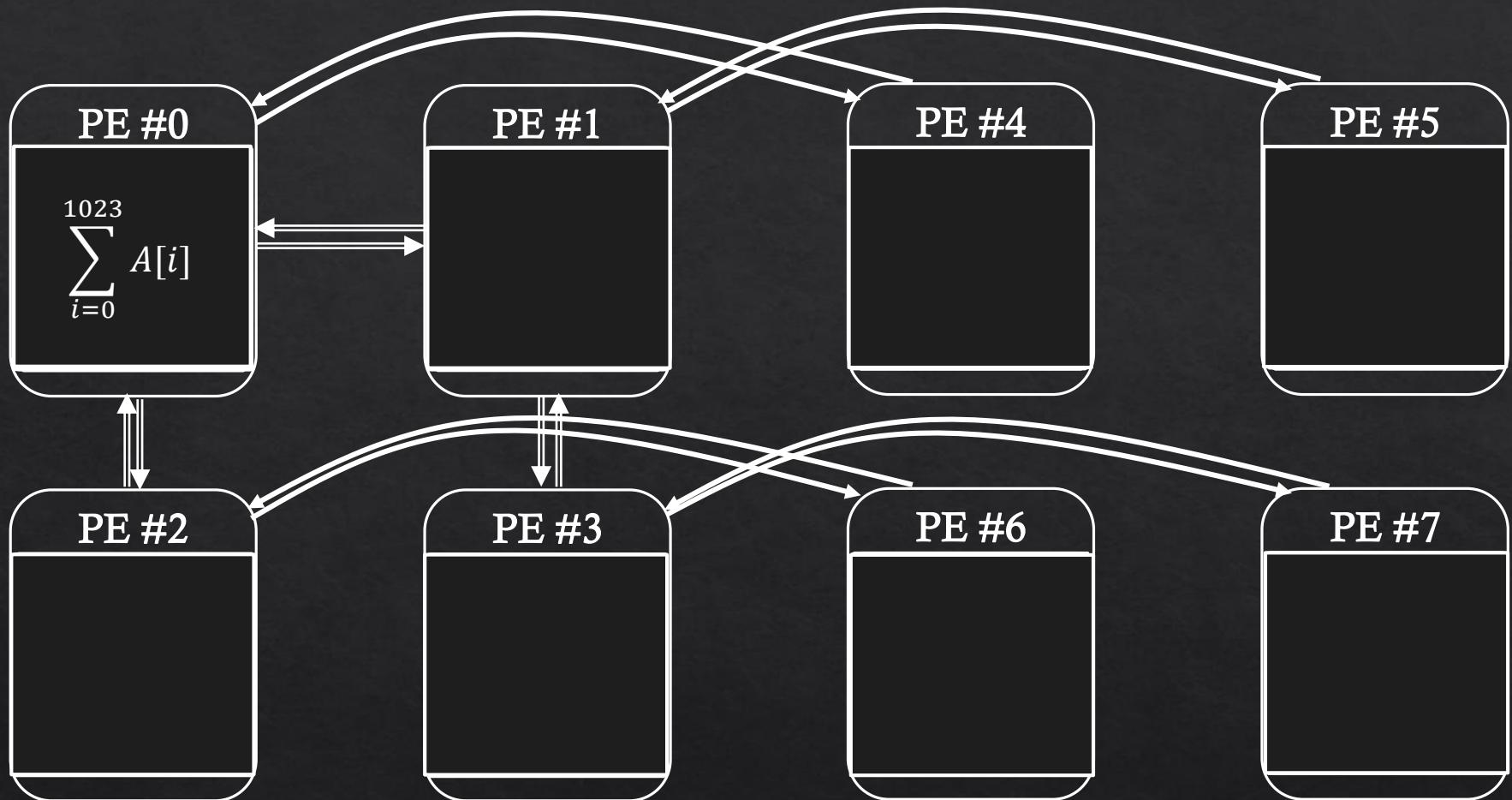
- ❖ 2 PEs ($p = 2$) ба 1024 ширхэг тоонууд ($n = 1024$):
 - ❖ $T(2,1024) = 3 + 511 + 3 + 1 = 518 \text{ сек}$
 - ❖ Speedup: $T(1,1024)/T(2,1024) = 1023/518 = 1.975$
 - ❖ Efficiency: $1.975/2 = 98.75\%$

Тоог нэмэх параллел алгоритм



- $T(4, 1024) = 3 \times 2 + 255 + 3 \times 2 + 2 = 269$ сек
- Speedup: $T(1, 1024)/T(4, 1024) = 1023/269 = 3.803$
- Efficiency: $3.803/4 = 95.07\%$

Тоог нэмэх параллел алгоритм



- $T(8,1024) = 3 \times 3 + 127 + 3 \times 3 + 3 = 148$ сек
- Speedup: $T(1,1024)/T(8,1024) = 1023/148 = 6.91$
- Efficiency: $6.91/8 = 86\%$

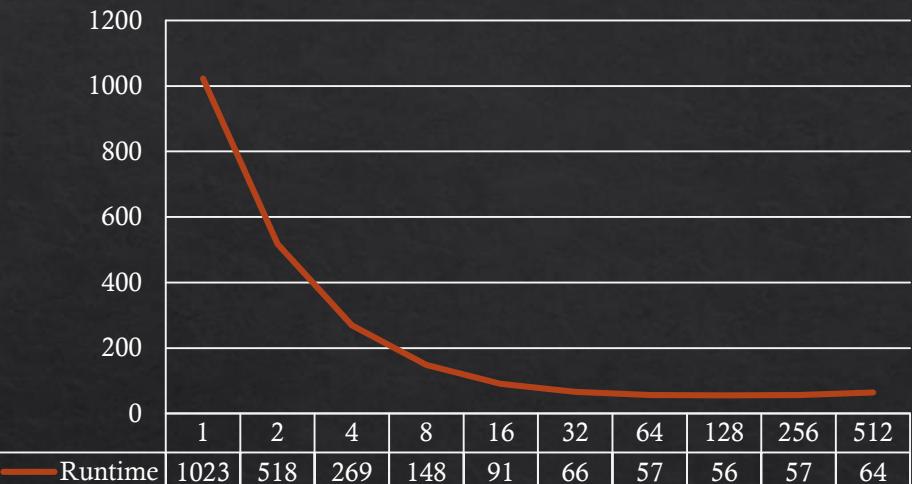
ШИНЖИЛГЭЭ

- ❖ $p = 2^q$ РЕ болон $n = 2^k$ ширхэг оролтын тонуудыг хэрэглэгсэн хугацааны шинжилгээ:
 - ❖ Өгөгдлийн тархалт: $3 \cdot q$
 - ❖ Дотоод нийлбэрийг тооцоолох: $n/p - 1 = 2^{k-q} - 1$
 - ❖ Хэсэгчилсэн дүнг цуглуулах : $3 \cdot q$
 - ❖ Хэсэгчилсэн дүнг нэмэх : q

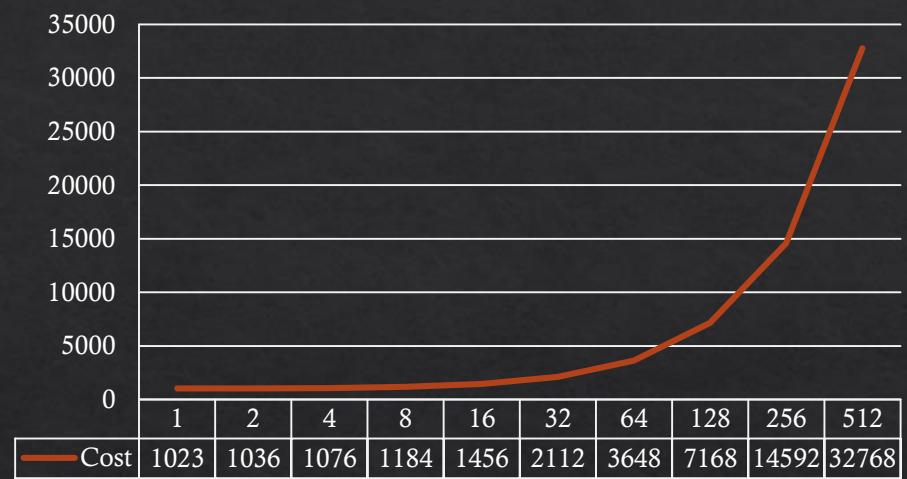
$$T(p, n) = T(2^q, 2^k) = 3q + 2^{k-q} - 1 + 3q + q = 2^{k-q} - 1 + 7q$$

Strong Scalability шинжилгээ. $n = 1024$

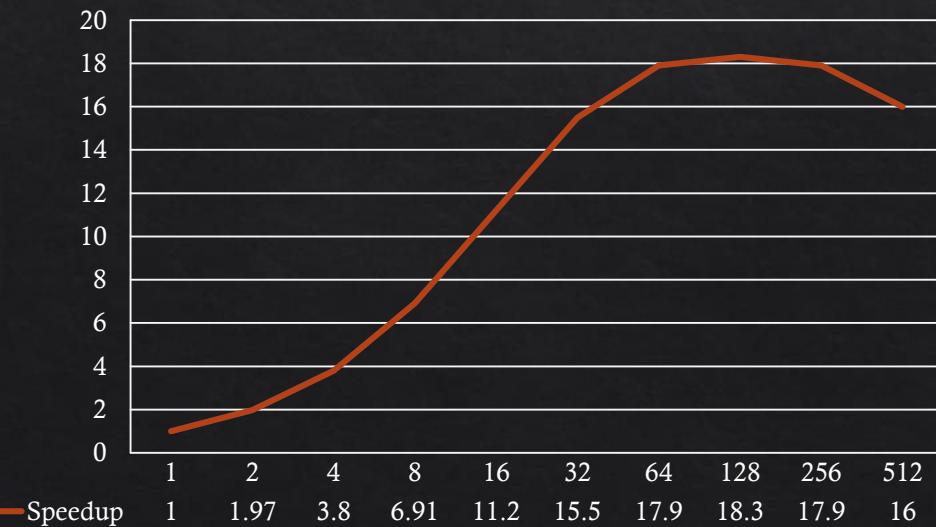
Runtime $\pi(1024, p)$



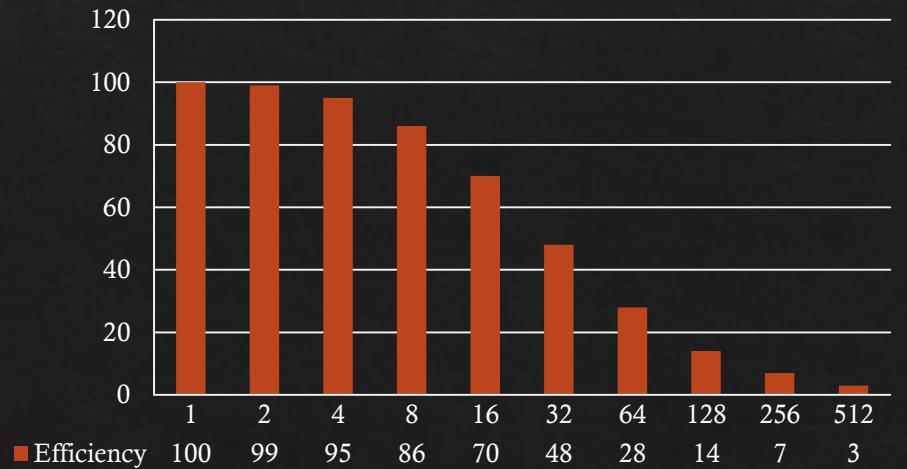
Cost = $\pi(1024, p) \times p$



Speedup = $\pi(1024, 1) / \pi(1024, p)$

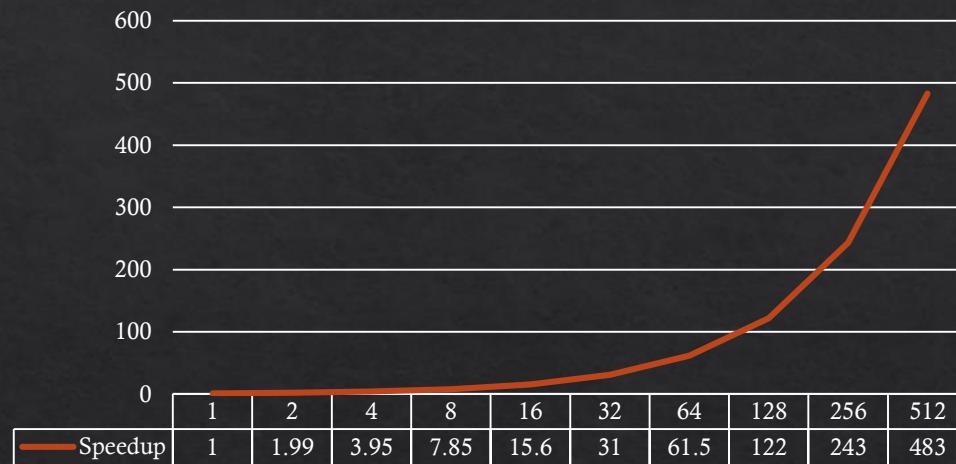


Efficiency (in %) = Speedup/ p

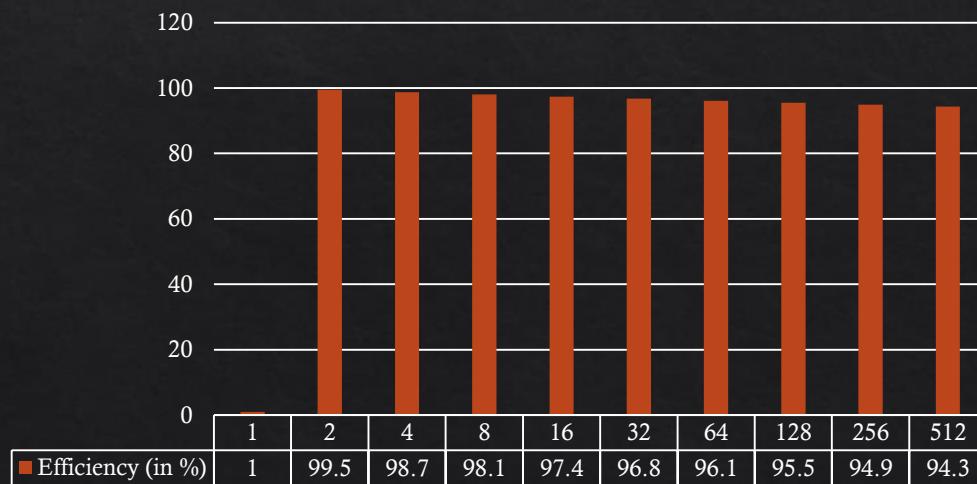


Weak Scalability шинжилгээ. $n = 1024 \times p$

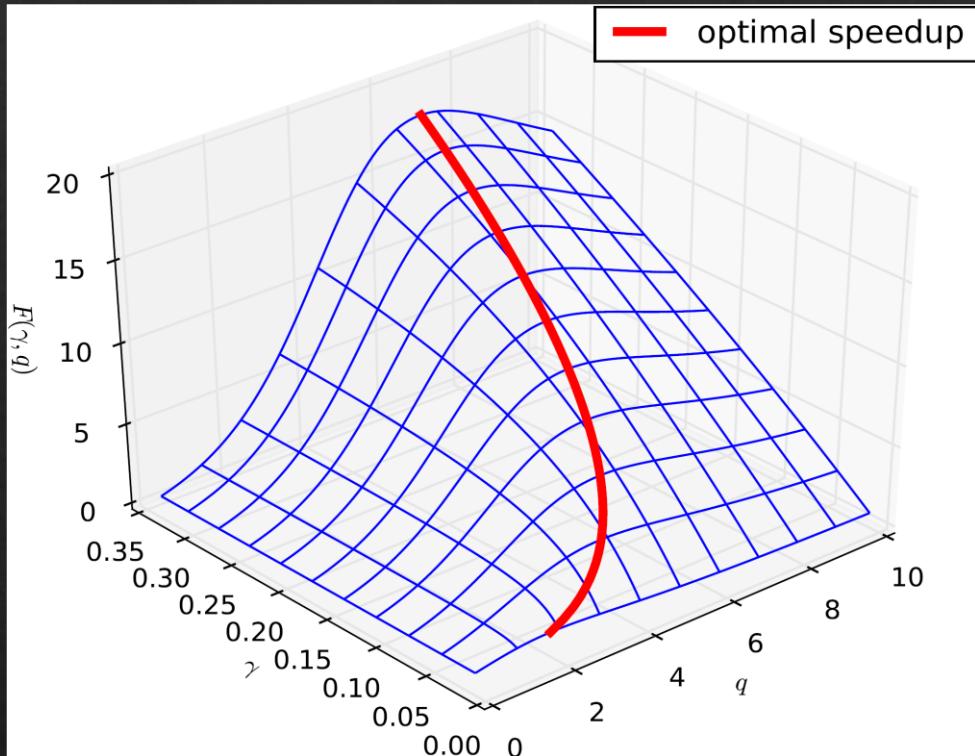
$$\text{Speedup} = T(1024 \times p, 1) / T(1024 \times p, p)$$



$$\text{Efficiency (in \%)} = \text{Speedup} / p$$



Compute-to-Communication Ratio



Нэг нэмэх үйлдлийн
хугацаа

$$\gamma = \frac{\alpha}{\beta}$$

Тоонуудын стакд
хандах хугацаа

$$T_{\alpha, \beta}(2^q, 2^k) = \beta q + \alpha(2^k - q - 1) + \beta q + \alpha q = 2\beta q + \alpha(2^k - q - 1 + q)$$

Speedup:

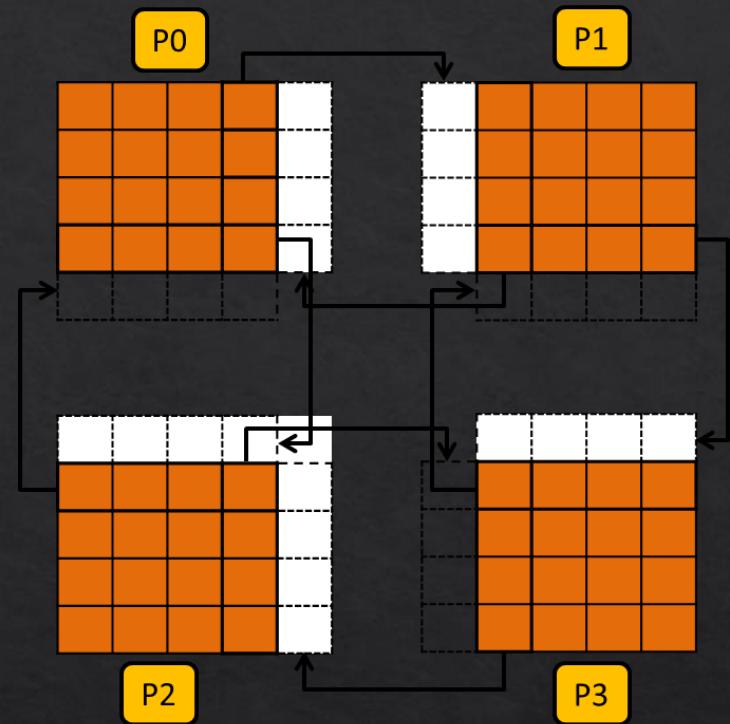
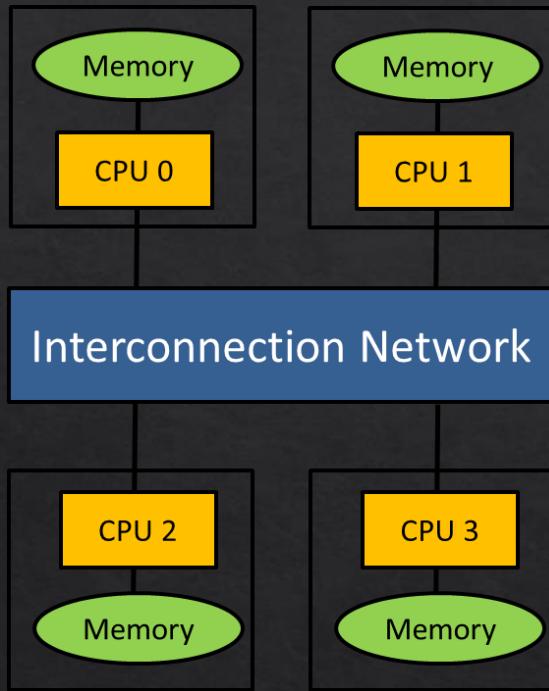
$$S_\gamma(2^q, 2^k) = \frac{\gamma(2^k - 1)}{2q + \gamma(2^{k-q} - 1 + q)}$$

Диаграм: $F(\gamma, q) := S_\gamma(2^q, 2^{10})$

Шинжилгээнээс гарах дүгнэлт

1. **Хурдсалт** нь асуудлын хэмжээ тогтмол үед тооцооллын нэгжүүдийн тоо болон computation-to-communication ratio хоёроос хамаарна.
 - ❖ Хурдсалт нь ерөнхийдөө тооцоолох нэгжийг нэмэгдүүлнэ. Хэтэрхий их болвол бас буурна.
 - ❖ Хурдсалтын оновчлол computation-to-communication харьцаанаас хамаарч байна. Харилцаа удаан байх тусам цөөхөн нэгж хэрэгэлэнэ.
2. **Параллелчлалын үр ашиг** нь асуудлын хэмжээ тогтмол үед тооцооллын нэгжүүдийн тоо болон computation-to-communication харьцаанаас хамаарна.

Хуваарилагдсан санах ойн системүүд

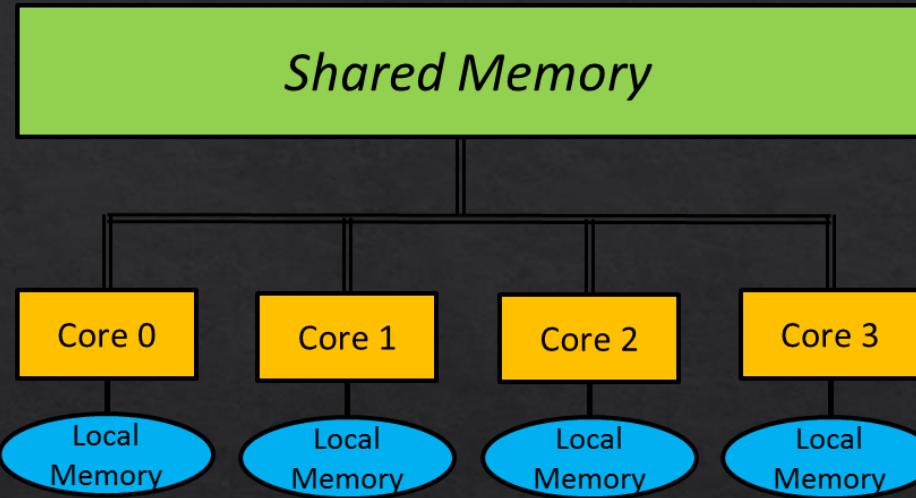


- ❖ Зангилаа бүр өөрийн гэсэн хувийн санах ойтой байдаг. Процессорууд сүлжээгээр мэдээлэлийг дамжуулж харилцдаг
 - ❖ Түгээмэл хэлнүүд: MPI (Жнь. MPI_Send, MPI_Recv, MPI_Bcast, MPI_Reduce)
 - ❖ PGAS хэл(Жнь. UPC++)

Interconnection network – Мэдээлэл солилцооны сүлжээ

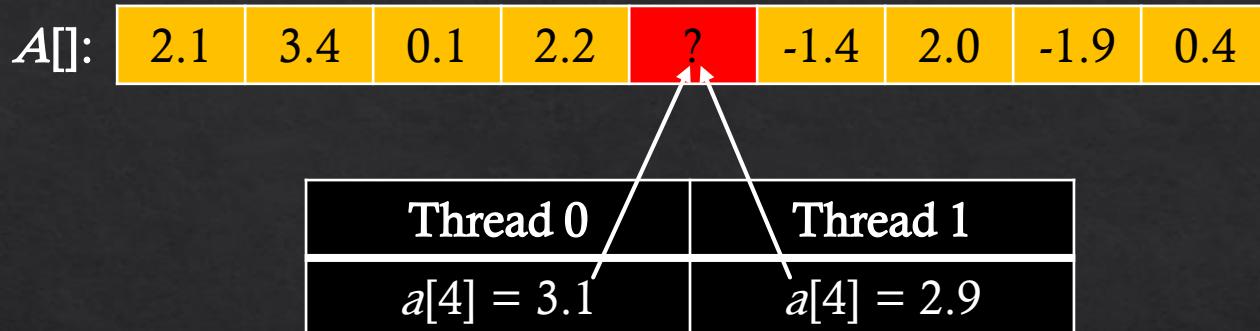
- ❖ Хуваарилагсан санах ойн системийн архитектурын маш чухал бүрэлдэхүүн хэсэг. Загвар нь :
 - ❖ point-to-point links
 - ❖ switching network
- ❖ *Network topology:* Олон программтай архитектурын чадамжийг тодорхойлно
- ❖ Хуваарилагсан санах ойн системийн оцлох жишээ
 - ❖ Тооцооллын кластер
 - ❖ Network-on-chip (NOC) архитектур.

Дундын санах ойн системүүд



- ❖ Бүх цөм нь дундын шугамаар дамжуулан үндсэн санах ойд хандана
 - ❖ Жнь. Зэрэгцээ цөмт CPU-суурилсан компьютерүүдийн бүх цөм нь үндсэн санах ойг дундаа ашигладаг
- ❖ Үндсэн санах ойгоос гадна цөм тус бүр нь жижиг санах ойг агуулж болно (жнь: L1-cache), үндсэн санах ойд хандахаас бага үнэлгээтэй
 - ❖ Орчин үеийн зэрэгцээ цөмт CPU-үүд нь кэш дэмждэг
 - ❖ *ccNUMA*: cache coherent non-uniform access architectures
- ❖ Түгээмэл хэлнүүд: C++11 multithreading, OpenMP, CUDA

Дундын санах ойн системүүд



- ◆ Системд зэрэгцээ ажиллаж байгаа *thread*-үүдээр параллелизм үүсгэгдэнэ
- ◆ Thread-н унших, бичих үйлдлээр дундын санах ойн руу өгөгдөл солилцно
- ◆ Race condition: Хоёр thread дундын хувьсагч руу нэгэн зэрэг хандах үед үүсдэг
 - ◆ Сэргийлэх програмчлалын техник : *mutexes, condition variables, atomics*
- ◆ Thread- үүсгэх нь процесс үүсгэхээс илүү хөнгөн, хурдан:
 - ◆ CreateProcess(): 12.76 ms
 - ◆ CreateThread(): 0.028 ms

Параллел програмын зохиомж

❖ Partitioning:

- ❖ Өгөгдсөн асуудлыг дэд хэсгүүдэд задрах шаардлагатай; жишээ нь. өгөгдлийн, даалгаврын болон загварын паралелизм гэсэн схемүүдээс сонгох хуваалт хийнэ

❖ Communication:

- ❖ Сонгосон хуваалтын схем нь шаардлагатай мэдээлэл солилцооны хэмжээ, төрлийг тодорхойлно

❖ Synchronization:

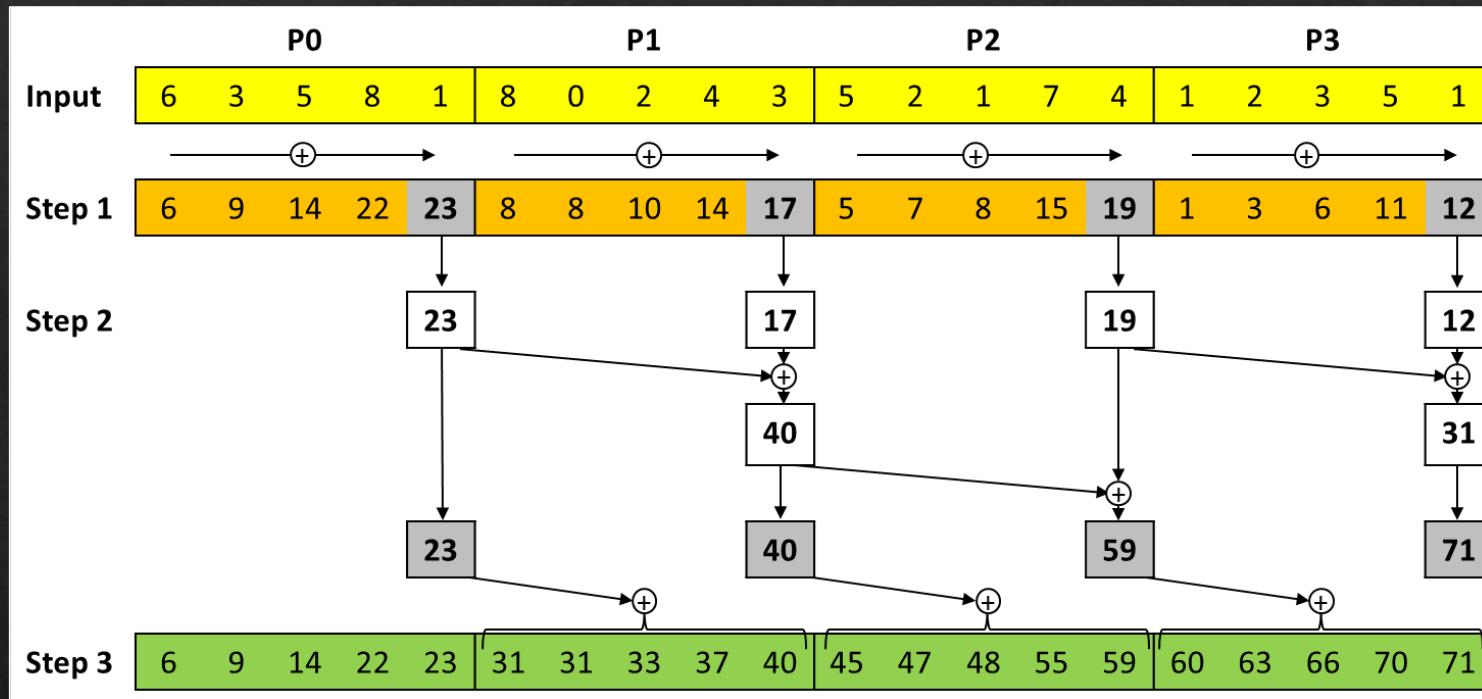
- ❖ Тохиромжтой байдлаар хамтран ажиллахын тулд thread эсвэл процессыг синхрончлох шаардлагатай байж болно

❖ Load Balancing:

- ❖ Ачааллыг тэнцвэржүүлэх, сүл зогсолтыг багасгахын тулд ажлыг thread болон процесст тэнцүү хуваах шаардлагатай

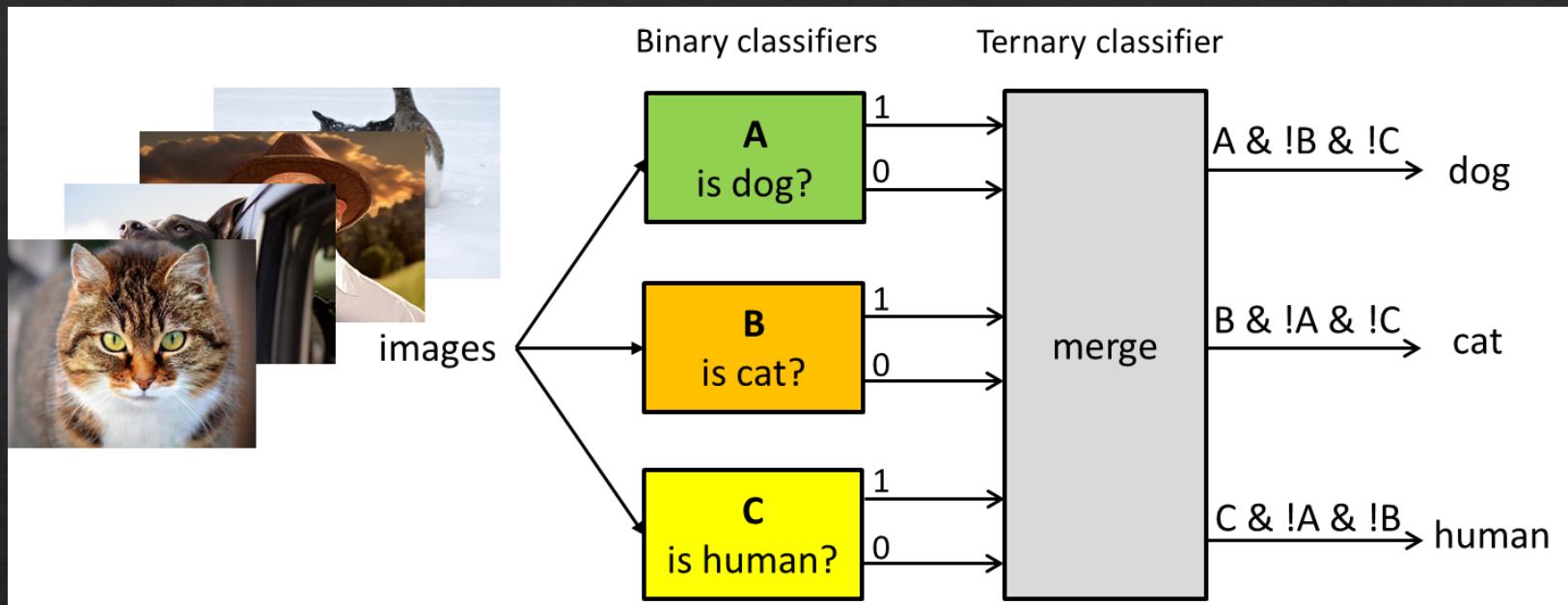
Параллелчлалын шинжилгээ (Prefix Sum)

```
for (i=1; i<n; i++) A[i]=A[i]+A[i-1];
```



- ❖ **Алхам 1:** процессор бүрийн дотоод нийлбэр
- ❖ **Алхам 2:** Дотоод массив бүрийн зөвхөн баруун талын утгыг ашиглан Prefix sum-г тооцоолно
- ❖ **Алхам 3:** Алхам2-д тооцоолсон утгыг зүүн хөршөөс авч дотоод массивийн элемент бүр дээр нэмэх

Partition-д хуваах



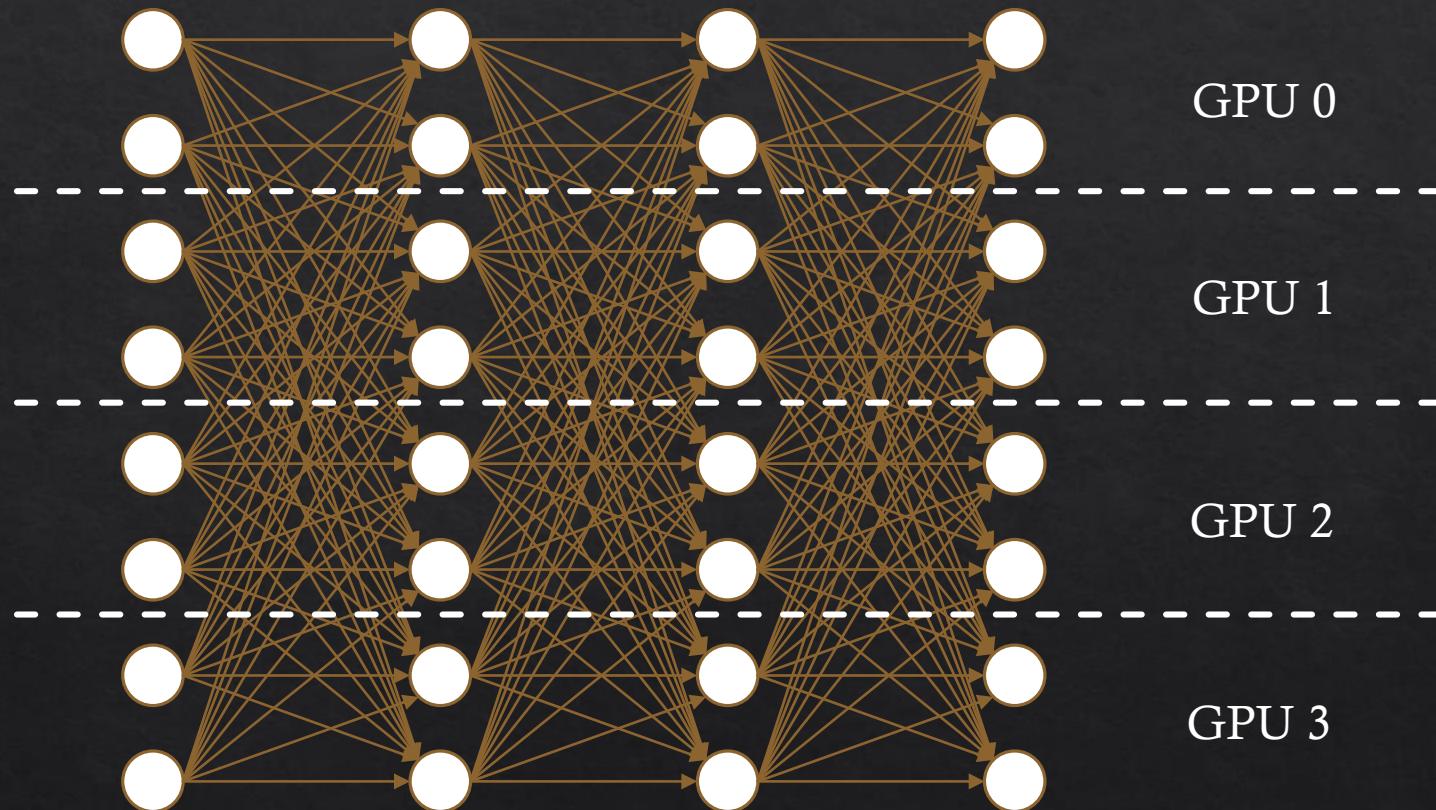
❖ Даалгаврын параллелизм

- ❖ Процесс бүрт өөр хоёртын ангилагч хуваарилагдсан (P_0, P_1, P_2)
- ❖ Бүх процесс нь өгөгдсөн ангилагчийг ашиглан зураг бүрийг ангилна.
- ❖ Зураг бурийн хоёртын ангиллын үр дүнг аль нэг процесс (Жнь. P_0) руу илгээж нэгтгэнэ
- ❖ Хязгаарлагдмал параллелизм ба ачааллын тэнцвэргүй байдал

❖ Өгөгдлийн параллелизм

- ❖ Оруулах зургийг хэд хэдэн багцад хувааж болно.
- ❖ Процесс нь хуваарилагдсан багцынхаа ангиллыг хийж дууссаход хуваарилагч шинэ багцыг динамикаар өгнө

ГҮН сургалтын Параллелизмын загвар



Баярлалаа.