Performance Analysis

σ(n) → inherently sequential computations
ρ(n) → potentially parallel computations
κ(n, p) → communication operations

SPEEDUP

$$speedup = \frac{sequential\ executio\ \text{time}}{parallel\ execution\ \text{time}}$$

$$\psi(n,p) = \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \frac{\varphi(n)}{p} + \varkappa(n, p)}$$

EFFICIENCY

$$speedup = \frac{\sigma(n)}{p \cdot \varphi(n)}$$

$$speedup = \frac{speedup}{processors}$$

AMDAHL'S LAW

$$f = \frac{\sigma(n)}{\sigma(n) + \cdot\ \varphi(n)} = share\ of\ parallel\ code$$

$$\psi \leq \frac{1}{f + \frac{1-f}{p}}$$

Example: 95% of the code can be executed in parallel

$$\psi \leq \frac{1}{0.05 - \frac{1 - 0.05}{8}}$$

Amdahl's law ignores the time needed for communication and thus themes to overestimate the speedup.

ISOEFFICIENCY METRIC

- compute total amount of overhead

$$T_0(n, p) = (p - 1) \cdot \sigma(n) + p \cdot \varkappa(n, p)$$

- Substitute into speedup formula

$$\psi(n,p) \leq \frac{p \cdot (\sigma(n) + \phi(n))}{\sigma(n) + \phi(n) + T_0(n,p)}$$

- assume efficiency stays constant

$$T(n,1) = \sigma(n) + \phi(n)$$

- determine relation between sequential execution time and overhead

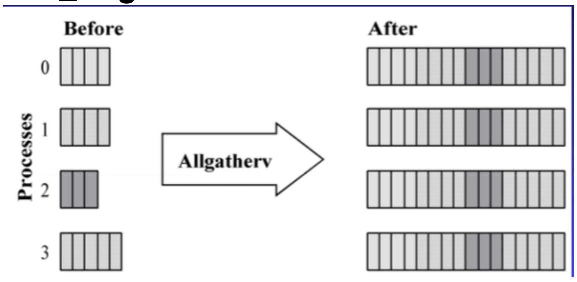
$$T(n,1) \geq C \cdot T_0(n,p)$$

This is the isoefficency relation

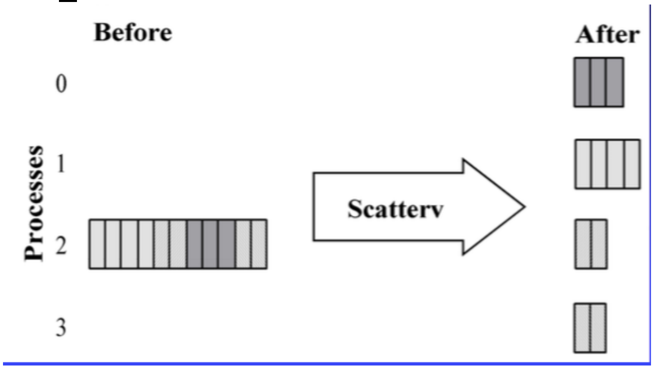
MPI

A communicator in MPI is a process group.

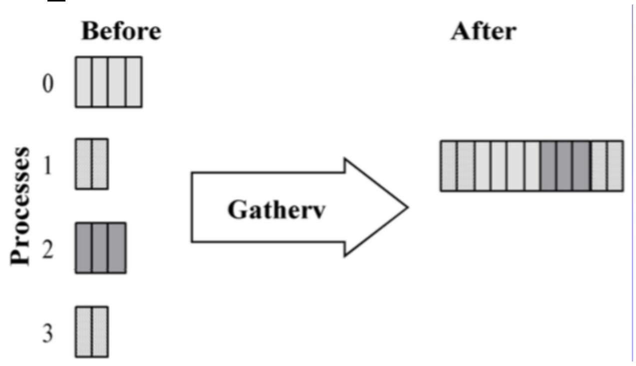
MPI_Allgatherv



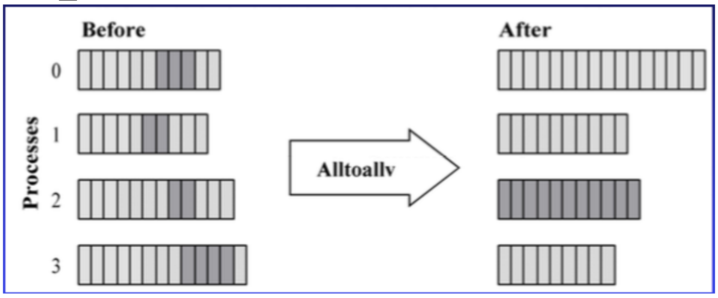
MPI_Scatterv



MPI_Gatherv



MPI_Alltoallv



MPI_Abort

Aborts the execution and terminates all MPI processes

MPI_Irec
Non-blocking receive. Using MPI_Request structure pointer to store the received data

MPI_Recv
Blocking receive

MPI_Wait
Waits until a message is received and stored in the given MPI_Request structure

MPI_Isend
Non-blocking send

MPI_Send
Blocking send

MPI_Probe
Blocks until a message is available to be received

Floyd's Algorithm

Finds the shortest paths for all pairs in a given matrix:

	A	B	C	D	E
A	0	6	3	6	4
B	4	0	7	10	8
C	12	6	0	3	1
D	7	3	10	0	11
E	9	5	12	2	0

Algorithm:

```
for k ← 0 to n-1
    for i ← 0 to n-1
        for j ← 0 to n-1
            a[i,j] ← min (a[i,j], a[i,k] + a[k,j])
        endfor
    endfor
endfor
```

Parallel execution possible by limiting n and setting the start values to something greater than 0.
Floyd's algorithm has a poor scalability because the memory needed increases with each process.

Manager/Worker MPI Program

Manager/Worker MPI programs have an early control flow split.
Workers are coordinated by manager, manager dispatches tasks to workers and collects results.
A separate worker comunicator can be created using MPI_Comm_split with the split_key MPI_UNDEFINED (global communicator is split, new communicator includes all processes but manager). Allows manger to send messages to all processes but to exclude himself.

Allows dynamic number of tasks to be assigned to a number of worker processes.No communication between workers. Variable task lengths.