2. Моделирование и анализ параллельных вычислений.

Модели вычислений.

Оценки эффективности параллельных алгоритмов.

Коммуникационная трудоемкость параллельных алгоритмов.

Эффективность параллельных вычислений

Жесткие требования к эффективности определены задачами (*экология, аэродинамика, геофизика, геном и др.*):

- исследуемые объекты – 3D,
- для приемлемой точности нужна сетка > 10^3 узлов,
- в каждом узле надо найти значения > 10 функций,
- при изучении динамики объекта определить его состояние в 10^2-10^4 моментах времени,
- на вычисление каждого результата в среднем приходится 10^2-10^3 арифметических действий,
- вычисления могут циклически повторяться для уточнения результата.

Зачем нужны модели и их анализ?

- Для разработки **эффективных** параллельных алгоритмов оценивают **эффективность использования параллелизма**:
  - эффективность **распараллеливания** конкретных выбранных **методов** выполнения вычислений,
  - **максимально** возможное **ускорение** процесса решения задачи (*анализируются **все** возможные способы выполнения вычислений*).
- Для этого нужны формальные **модели** вычислений

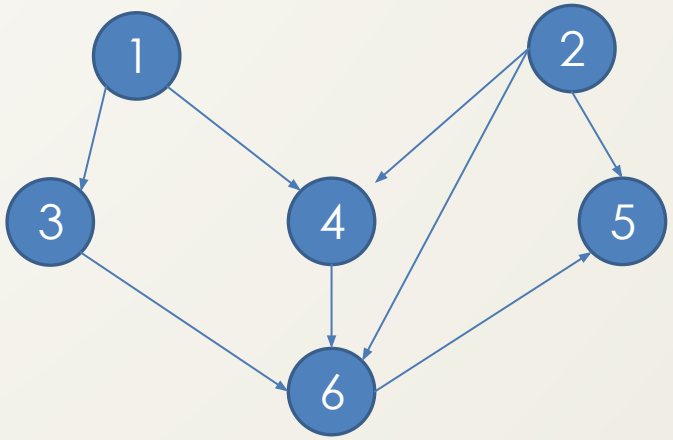
Модель вычислений

- **Граф «операции – операнды»** описывает алгоритм вычислений на уровне операций и информационных зависимостей.
- **Предположения упрощенной модели**:
  - **Время выполнения** всех вычислительных операций = **const = 1**.
  - **Передача данных** между PU выполняется **мгновенно** (например, в системе в общей памятью)

можно не учитывать коммуникационную трудоемкость алгоритмов

Определение графа «операции-операнды» (граф О-О)

- Граф О-О **G = (V , R)** – ***ациклический* ориентированный** (направленный) граф - в нем отсутствуют пути, начинающиеся и кончающиеся в одной и той же вершине.



Определение графа «операции-операнды» (граф О-О)

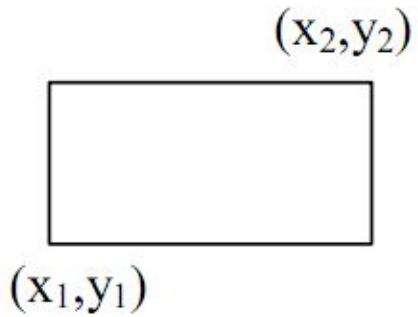
G = (V , R)

V – множество вершин графа, представляющих **выполняемые операции алгоритма.**

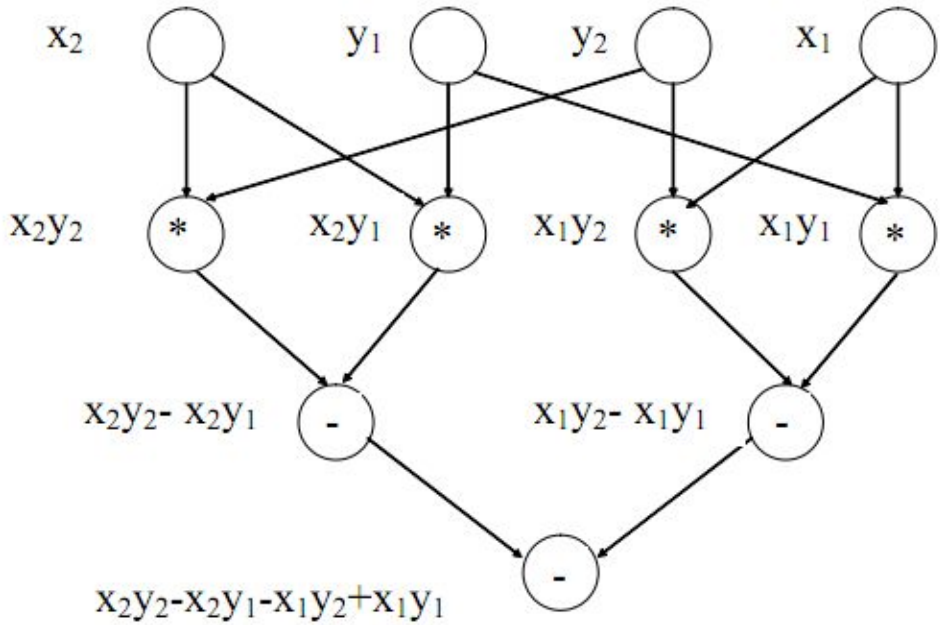
R – множество дуг графа, определяющих **последовательность операций**.

- Дуга **r(i,j)** принадлежит графу G только, если операция j использует результат выполнения операции i
- Вершины **без входных дуг** могут использоваться для указания операций **ввода**
- Вершины **без выходных дуг** – для указания операций вывода.

Пример модели вычислений (S прямоугольника)



$$S = (x_2-x_1)(y_2-y_1) = x_2y_2-x_2y_1-x_1y_2+x_1y_1$$



Комментарии к модели

- Можно выбрать иную схему вычислений и построить другую модель.
- Операции, между которыми нет пути, можно выполнять параллельно (сначала все «*», потом «-»)

Оценки трудоемкости параллельных алгоритмов

- **Speedup** - **ускорение** за счёт параллельного выполнения на N процессорах (потоках)

$a(N) = T(1) / T(N)$

- **Efficiency - эффективность** системы из N процессоров

$E(N) = a(N) / N$

- **Scalability** - **масштабируемость** системы (возможность ускорения вычислений пропорционально числу процессоров,
  т.е. линейное ускорение)

$a(N)=N$

- $a(N)>N$ – суперлинейное ускорение

Граф О-О и показатели эффективности для типовых задач

Алгоритмы вычисления сумм

Общая задача - нахождение **частных** сумм последовательности числовых значений

$$S_k = \sum_{i=1}^{k} x_i, 1 \le k \le n$$

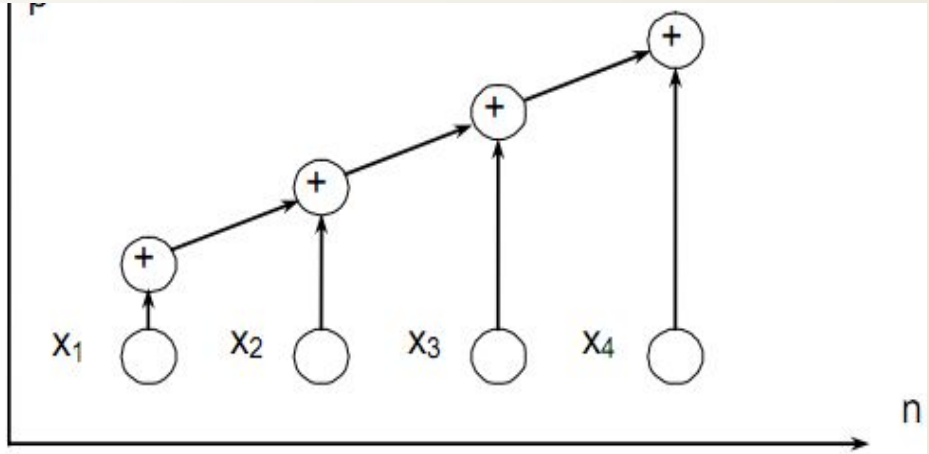
Частная задача - нахождение **общей** суммы последовательности числовых значений

$$S = \sum_{i=1}^{n} x_i$$

Последовательный алгоритм

$S=0;\ S+=x_1;\ \ldots$

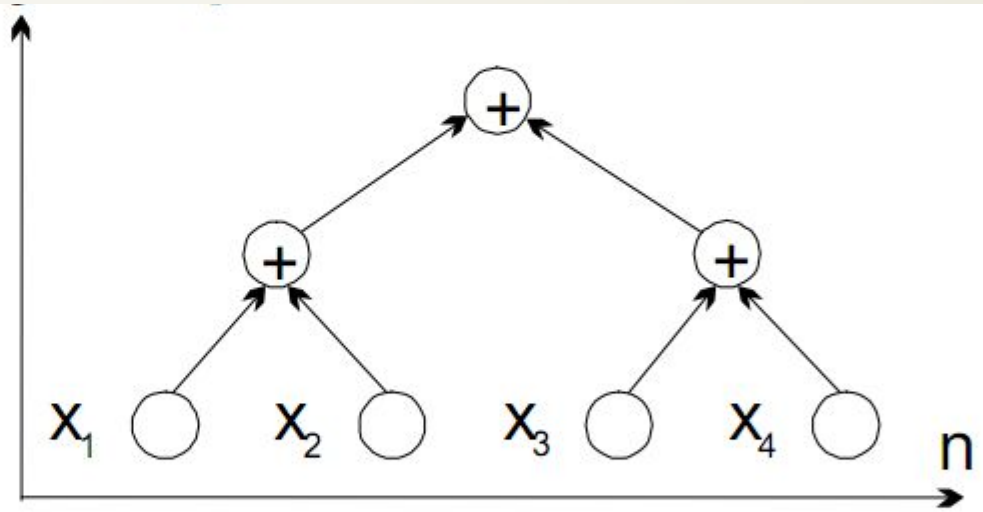
$$S = \sum_{i=1}^{n} x_i, \qquad G_1 = (V_1, R_1)$$



Возможно только последовательное выполнение, распараллеливание данного алгоритма выполнить нельзя.

Каскадная схема

$$G_2 = (V_2, R_2)$$



Возможно распараллеливание алгоритма суммирования

Оценка эффективности

- Количество итераций (уровней) каскадной схемы:

$$k = \log_2 n$$

(на рисунке при $n=4, k=2$)

- Общее количество операций суммирования по всем уровням **без распараллеливания** (равно количеству суммирований для последовательной схемы):

$$K_{посл} = n/2 + n/4 + \ldots + 1 = n - 1$$

- Общее количество операций суммирования **с распараллеливанием** (равно количеству итераций):

$$K_{пар} = \log_2 n$$

Оценка эффективности

- Время выполнения на 1 процессоре (равно количеству операций):

$$T_1 = K_{посл}$$

- Время выполнения на p процессорах (равно количеству итераций):

$$T_p = K_{пар}$$

- Ускорение:

$$a(p) = T_1/T_p = (n - 1)/\log_2 n$$

- Эффективность:

$$E(p) = a(p)/p = (n - 1)/(p * \log_2 n)$$

Для реализации вычислительной схемы необходимо $n/2$ процессоров

$$E(p) = a(p)/p = (n - 1)/((n/2) * \log_2 n)$$

$$\lim E(p) = 0 \text{ при } p \to \infty$$

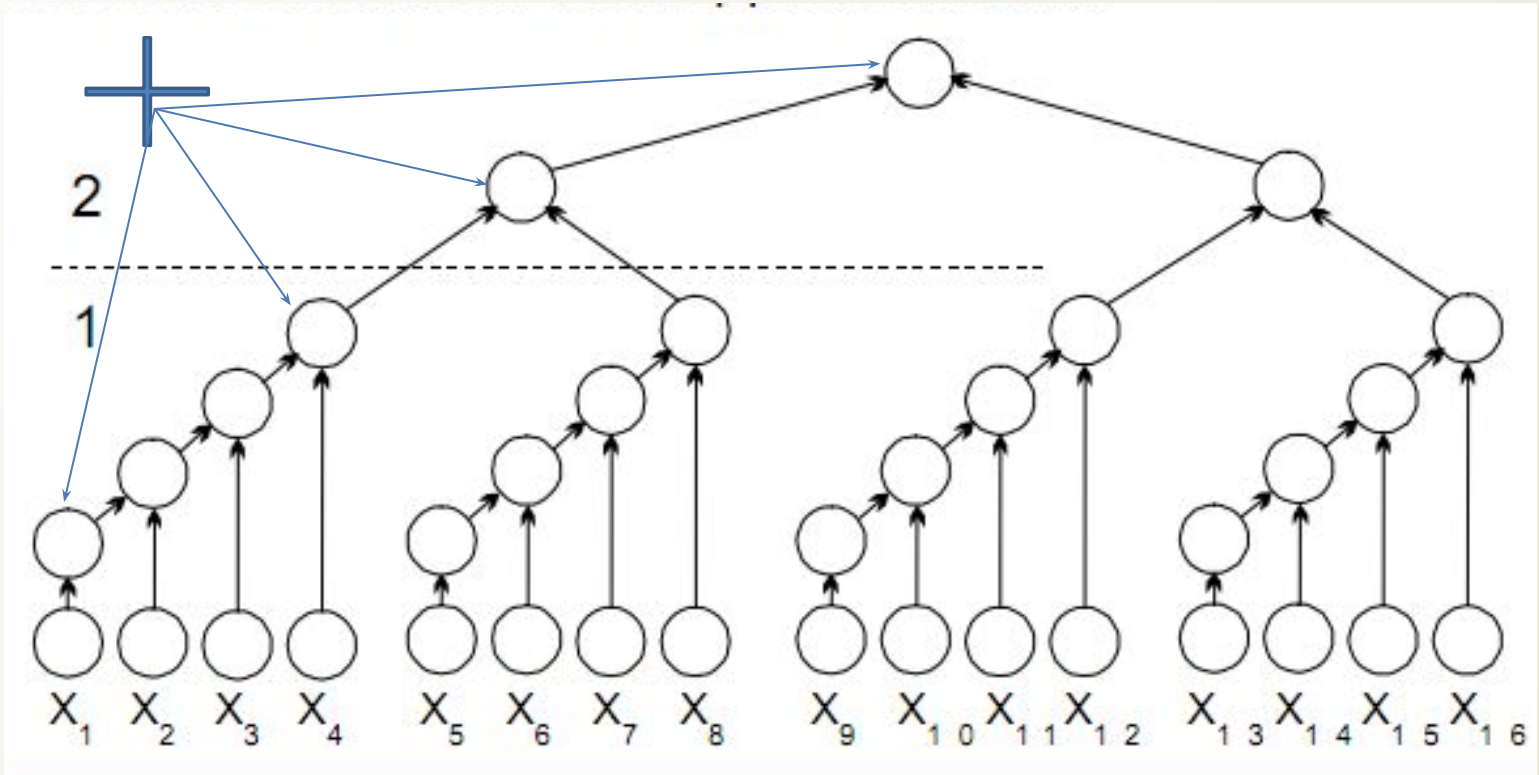
Улучшение каскадной схемы

- **Цель** – асимптотически **ненулевая** эффективность.
- **Суть** алгоритма:
  - все суммируемые значения подразделяются на ($n/\log_2 n$) групп, в каждой из которых содержится ($\log_2 n$) элементов;
  - для каждой группы вычисляется (параллельно) сумма значений при помощи последовательного алгоритма суммирования;
  - для полученных ($n/\log_2 n$) сумм отдельных групп применяется обычная каскадная схема.

Улучшение каскадной схемы

- **Пример при $n=16$**:
  - все суммируемые значения подразделяются на $(n/\log_2 n)=4$ группы, в каждой из которых содержится $(\log_2 n)=4$ элемента;
  - для каждой группы вычисляется (параллельно) сумма значений при помощи последовательного алгоритма суммирования;
  - для полученных $(n/\log_2 n)=4$ сумм отдельных групп применяется обычная каскадная схема.

Модифицированная каскадная схема



Оценка эффективности

- **Для этапа 1**
  - число параллельных операций $k_1 = \log_2 n$
  - необходимое число процессоров $p_1 = n/\log_2 n$
- **Для этапа 2**
  - число параллельных операций $k_2 = \log_2(n/\log_2 n) \le \log_2 n$
  - необходимое число процессоров $p_2 = p_1/2 = (n/\log_2 n)/2$

- Время выполнения на $p = p_1$ процессорах:

$$T_p = k_1 + k_2 \le 2 \log_2 n$$

- Ускорение (**меньше** в 2 раза):

$$a(p) = T_1/T_p = (n-1)/2 \log_2 n$$

- Эффективность (асимптотически **ненулевая**):

$$E(p) = a(p)/p = (n-1)/2 p\log_2 n = (n-1)/2n$$

$$\lim E(p) = 0.5 \text{ при } p \to \infty$$

Выводы

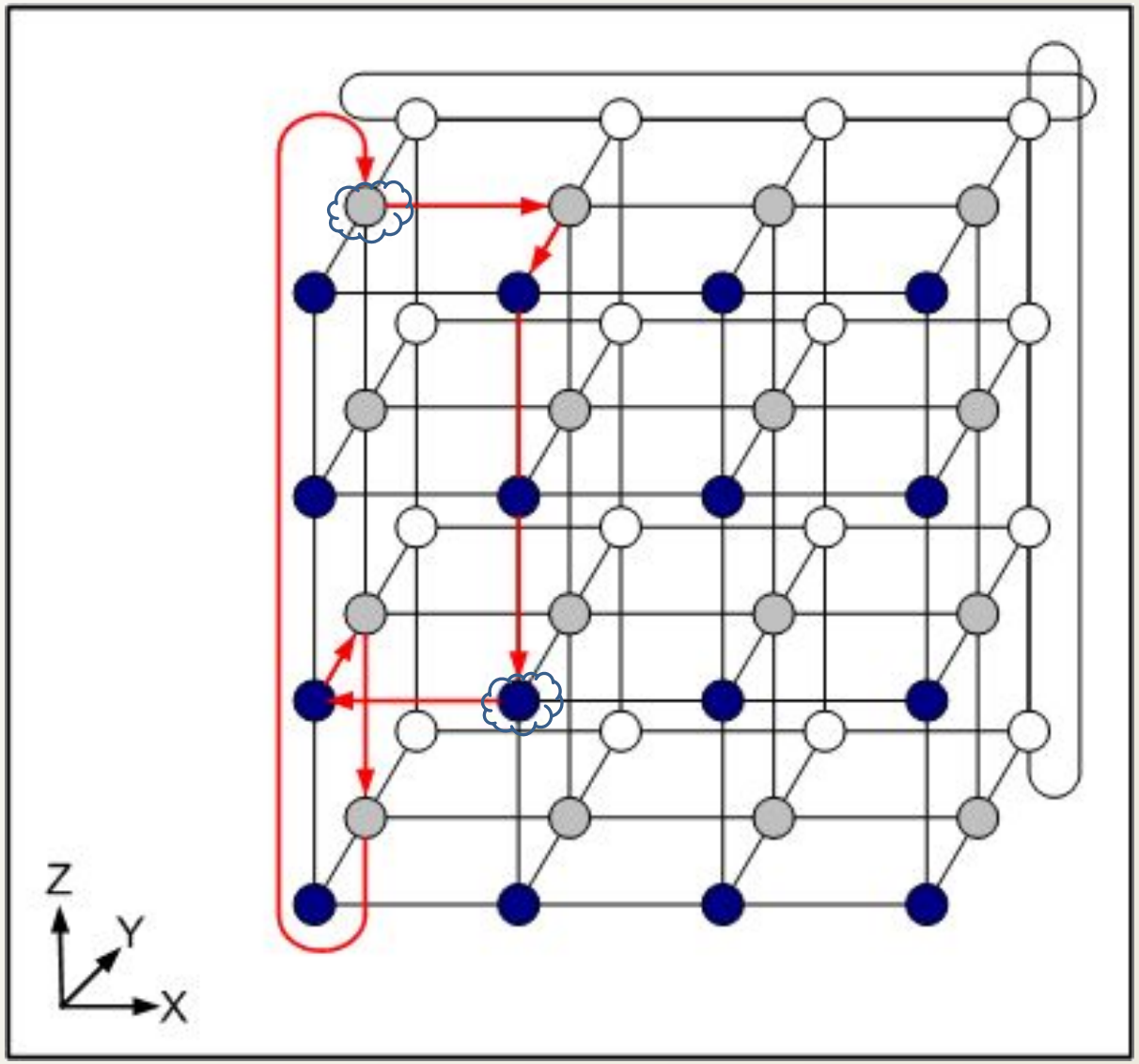
- Неочевидность приемов распараллеливания
- Неочевидность оценок эффективности
- Возможность разных вариантов параллельных схем
- Наглядность модели на графах

Передача данных. Коммуникационная трудоемкость алгоритмов

- В рассмотренных оценках **не учтены затраты времени на передачу данных.**
- Основа для характеристики передачи данных **– алгоритмы маршрутизизации (АМ)**.
- АМ определяет **путь передачи данных** от CPU1 (источника сообщения) до CPU2 (адресата доставки).
- **Классификация АМ**:
  - **Оптимальные** (определяют **кратчайшие** пути передачи данных) и **неоптимальные** АМ.
  - **Адаптивные** (учитывают **загрузку** каналов связи) и **неадаптивные** АМ.

Пример оптимальных АМ

- Алгоритмы, основанные на **покоординатной** маршрутизации (*dimension ordered routing*) – **поочередный** поиск путей передачи данных для **каждой размерности** топологии сети.
- Пример: **алгоритм XY-маршрутизации** для решетки:
  - Передача данных по **горизонтали** до пересечения с вертикалью CPU2
  - Передача данных по **вертикали и т.д.**


Z
Y
X

Характеристики коммуникационной составляющей длительности выполнения параллельного алгоритма в МВС

- **Время передачи данных** определяют:
  - **Время начальной подготовки** сообщения для передачи, поиска маршрута в сети – $t_н$
  - **Время передачи служебных данных** (*заголовок сообщения, диагностический блок*) между соседними CPU (имеющими между собой физический канал передачи данных) – t_c
  - **Время передачи одного байта по одному каналу** (*определяется полосой пропускания каналов сети*) – $t_к = 1/R$, где R - ширина полосы

Методы передачи данных

1. **Метод передачи сообщений как неделимых блоков информации** (*store-and-forward routing, SFR*):
   - CPU1
     - Готовит данные (сообщение) для передачи
     - Определяет CPU2 для пересылки (промежуточный)
     - Запускает операцию пересылки данных
   - CPU2
     - Принимает **полностью** все пересылаемые данные
     - Выполняет пересылку далее по маршруту

Время пересылки m байт по маршруту длины l (через l узлов) :

$$t_{пд} = t_н + (mt_к + t_с)l$$

Для «длинных» сообщений, где можно пренебречь пересылкой служебных данных:

$$t_{пд} = t_н + mt_к l$$

Методы передачи данных

2. **Метод передачи пакетов** – сообщение состоит из блоков информации (пакетов) (*cut-through-routing, CTR*)

- CPU1
  - Готовит данные (сообщение) в виде пакетов для передачи
  - Определяет CPU2 для пересылки (промежуточный)
  - Запускает операцию пересылки пакетов
- CPU2
  - Принимает **пакет**
  - Выполняет пересылку далее по маршруту как только получил и обработал заголовок (учитывает t_c)

Время пересылки m байт по маршруту длины l :

$$t_{пд} = = t_н + mt_к + t_c l$$

Преимущества и недостатки CTR

- Ускоряет пересылку данных.
- Снижает потребность в памяти для хранения пересылаемых данных и организации приема-передачи сообщений.
- Для передачи могут использоваться одновременно разные коммуникационные каналы.
- Требует разработки более сложного аппаратного и программного обеспечения сети.
- Может увеличить накладные расходы (время подготовки и время передачи служебных данных),
- При передаче пакетов возможно возникновение конфликтных ситуаций.

Классификация операций передачи данных в МВС

- **передача** данных (сообщений):
  - между двумя CPU сети,
  - от одного CPU всем остальным CPU сети,
  - от всех CPU всем CPU сети,
  - то же для различных наборов данных;
- **прием** данных (сообщений):
  - на один CPU от всех CPU сети,
  - на каждом CPU от всех CPU сети,
  - то же для различных сообщений.

Оценки трудоемкости для различных топологий

Топология	**Диаметр**
Граф	1
Звезда	2
Линейка	p - 1
Кольцо	p/2
Решетка (2D)	2(√p - 1)

- **Диаметр** – определяет время передачи данных, **max расстояние** между 2 CPU сети (*расстояние равно величине кратчайшего пути*).
- Для оценки нужно:
  - Определить алгоритм пересылки.
  - В формулы вместо l подставить значение диаметра