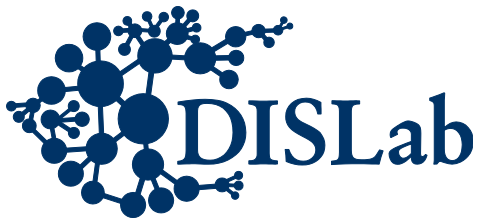
Спецкурс «Параллельная обработка больших графов»

Лекция 5

А.С. Семенов

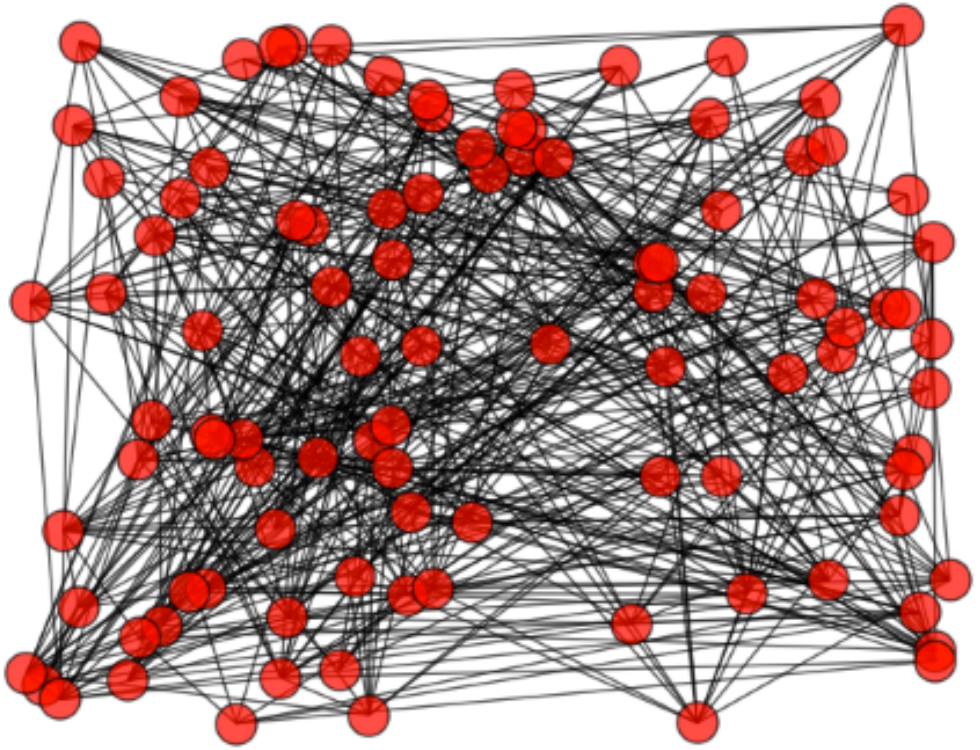
dislab.org



Виды графов

Виды графов. Случайные графы

- Random, Random Uniform, Erdos Renyi
- N вершин, M ребер, k – средняя связность вершины



Виды графов. Степенной закон

- WWW, Социальные сети, Биоинформатика
- Графы small-world

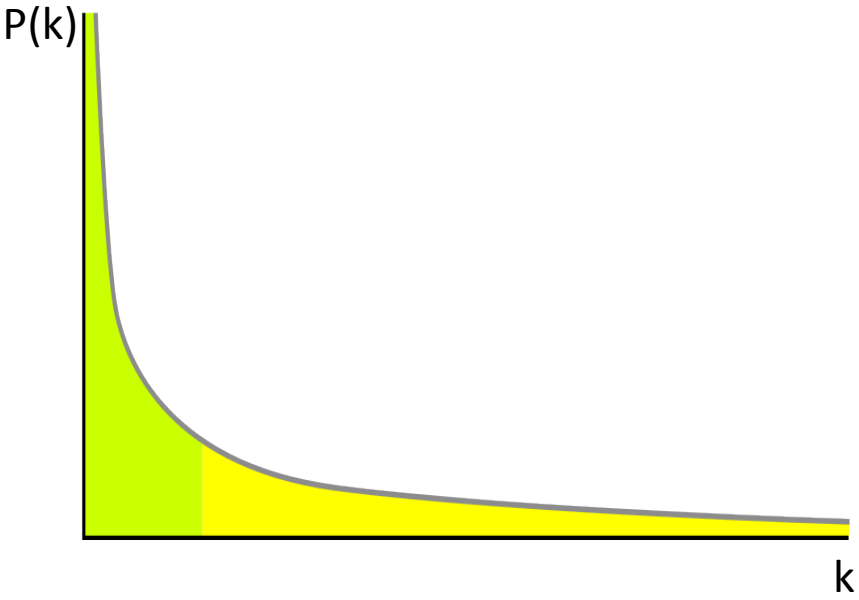
$L \sim \log N$

- scale-free – графы,

доля $P(k) \sim k^{-tau}$, $2 < tau < 3$

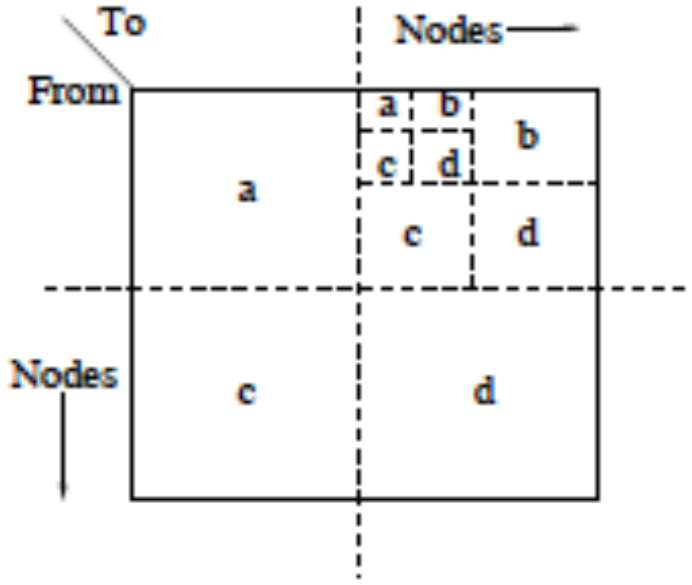
k – связность вершины

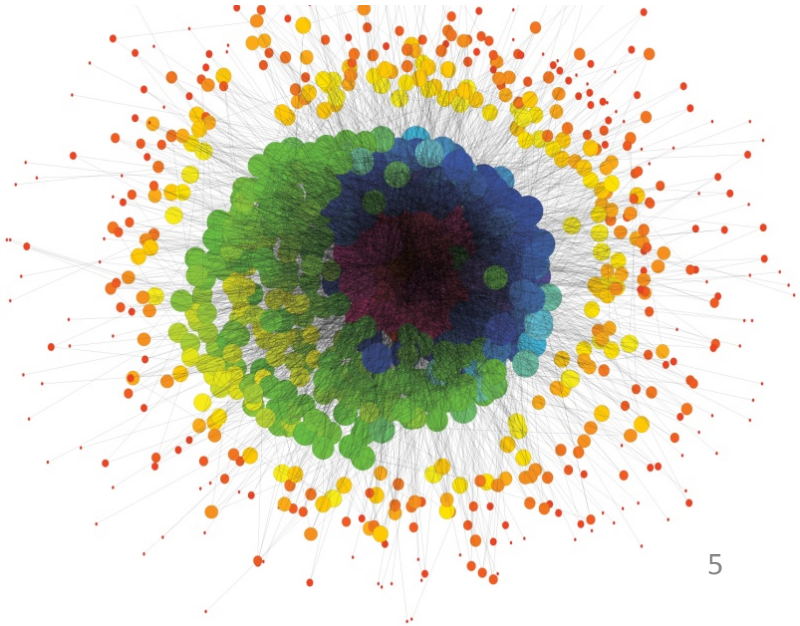
$L \sim \log \log N$



Виды графов. RMAT-граф

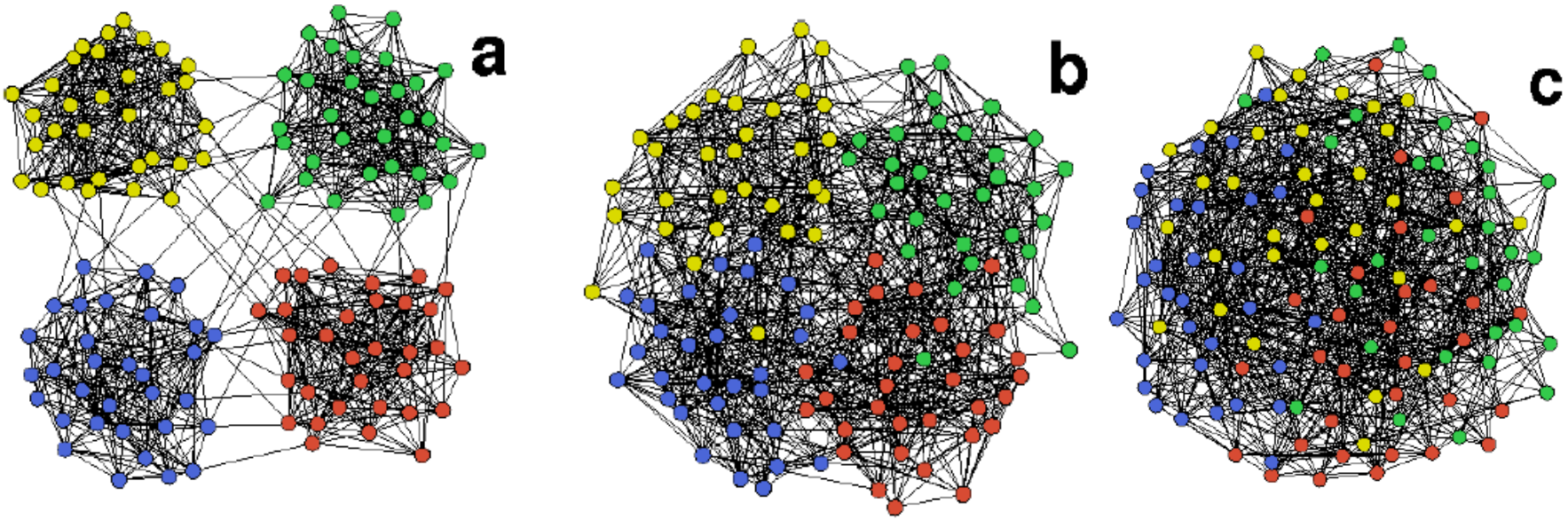
- a+b+c+d = 1
- Сообщества:
  - a и d – сообщества
  - b и c – связи между ними
  - наличие «подсообществ»
- может быть scale-free при a>=d
- случайная перестановка вершин





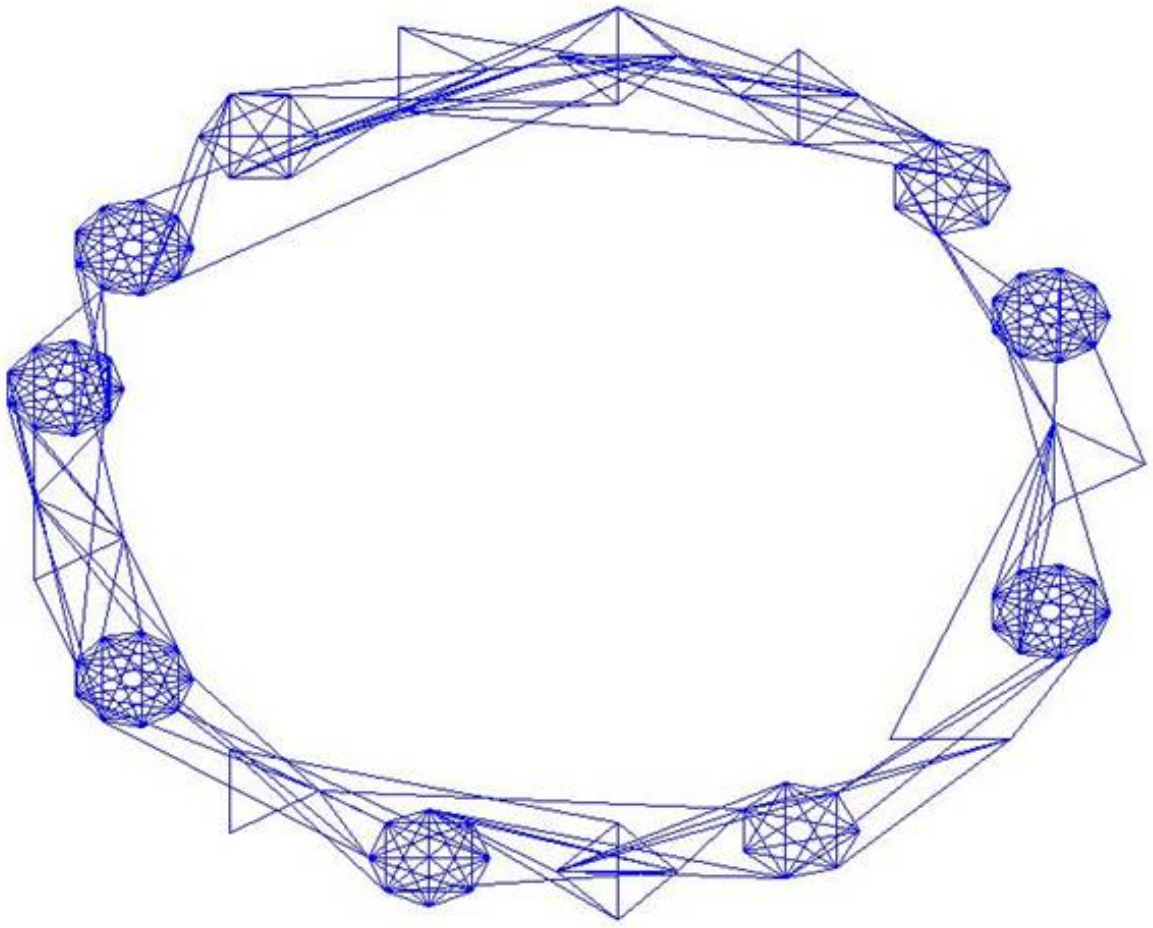
Виды графов. LFR*-граф

- **Параметры**:
  - mu ∈ [0;1], показывает количество связей вне сообщества
  - com_tau – показатель степени в законе распределения размеров сообществ
  - deg_tau – показатель степени в законе распределения степеней вершин



Виды графов. SSCA2-граф

- Равномерное распределение случайных параметров
- случайная перестановка вершин

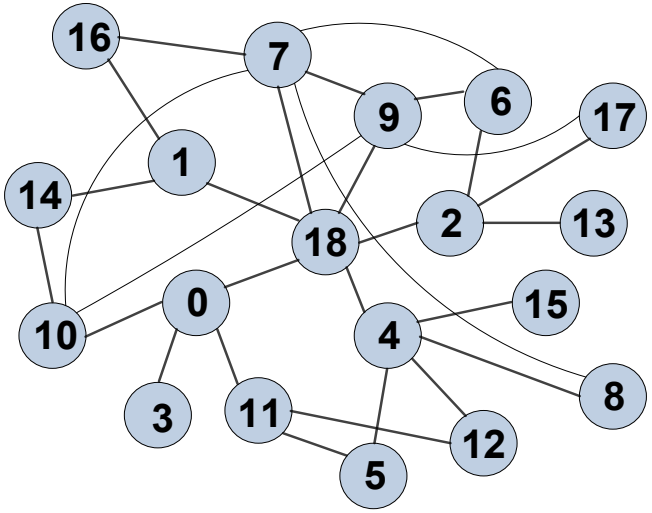


Проблемы и подходы к решению задач обработки графов в рамках одного вычислительного узла

Проблемы анализа больших графов

- **Data-driven computations.** Зависимость вычислений от данных (топологии графа). Невозможность применения методов статического распараллеливания вычислений.
- **Unstructured problems.** Работа с нерегулярными, неструктурированными данными, трудность распараллеливания.
- **Poor locality.** Низкая пространственно-временная локализация обращений к памяти.
- **High data access to computation ratio.** Преобладание команд доступа к памяти над командами выполнения арифметических операций.

Представление графа



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0				1							1	1							1
1															1		1		1
2							1							1				1	1
3	1																		
4						1			1				1			1			1
5					1							1							
6			1					1		1									
7							1		1	1	1						1		1
8					1			1											
9							1	1			1							1	1
10	1							1		1					1				
11	1					1							1						
12					1							1							
13			1																
14		1									1								
15					1														
16		1						1											
17			1							1									
18	1	1	1		1			1		1									

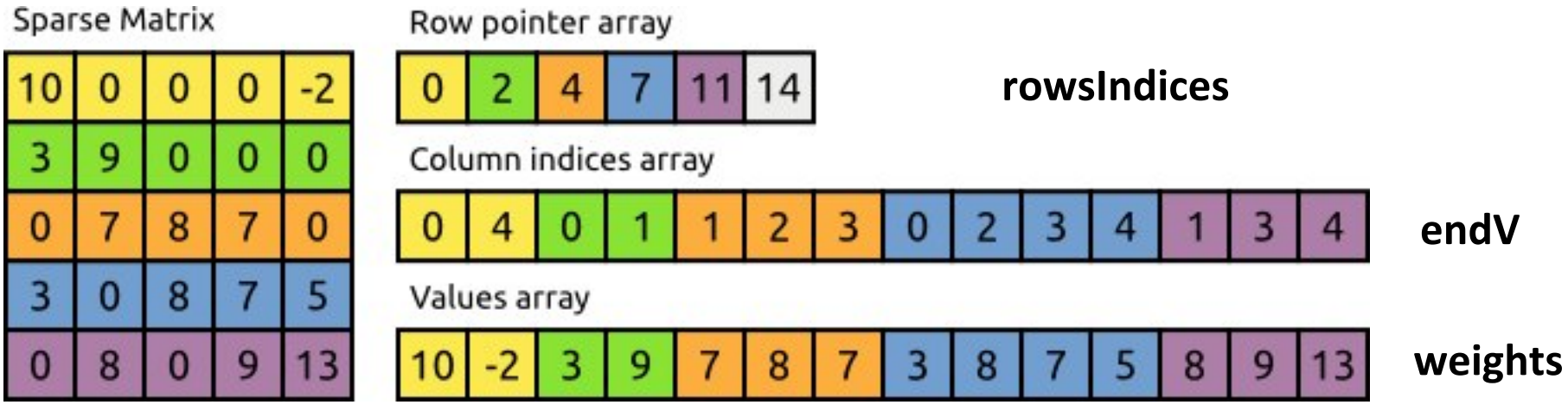
Форматы представления разреженных матриц

- **Доля ненулевых элементов мала**

Можно хранить только позиции и значения ненулевых элементов

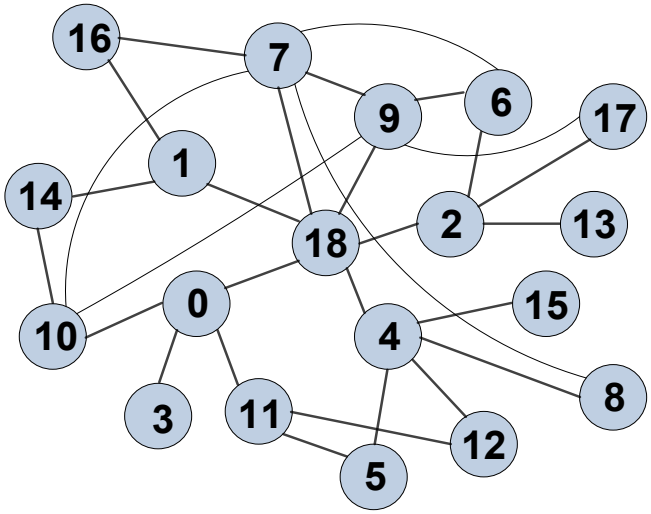
- **Compressed Row Storage (CRS)**
- **Coordinate list (COO)**
- **DIA**
- **ELLPACK**
- **SELLPACK**
- **Оптимизированный под задачу**

Внутреннее представление Compressed Row Storage (CRS)

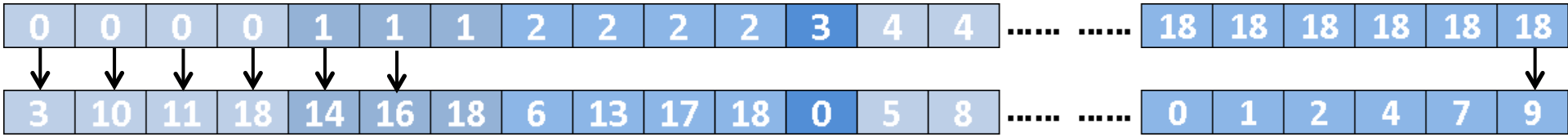



```
for (int u = 0; u < G->n; u++) {
   for (int j = G->rowsIndices[u]; j < rowsIndices[u+1];
   j++) {
    const int v = G->endV[j];
    const int w = G->weights[j];
    // обработка ребра u->v
   }
}
```

Coordinate list (COO)



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0				1							1	1							1
1															1		1		1
2							1							1				1	1
3	1																		
4						1			1				1			1			1
5					1							1							
6			1					1		1									
7							1		1	1	1						1		1
8					1			1											
9							1	1			1							1	1
10	1							1		1					1				
11	1					1							1						
12					1							1							
13			1																
14		1									1								
15					1														
16		1						1											
17			1							1									
18	1	1	1		1			1		1									



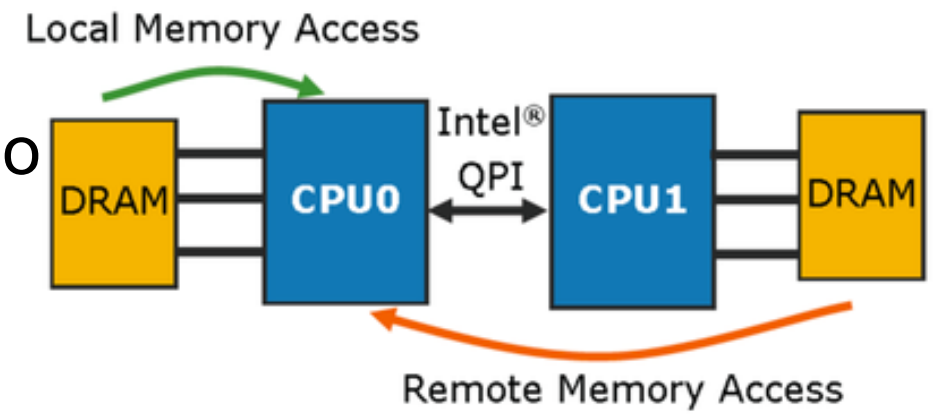
Общие методы программной оптимизации для SMP x86 систем

Архитектура вычислительного узла

- 2 сокета с Intel Xeon E5-2683 v3 2.00 GHz
- Включен режим Turbo до 3 GHz
- кэш 35 МБ
- NUMA
- 14 ядер на сокет
- Hyper-Threading (в сумме 56 потоков)
- QPI 9.6 GT/s
- 64 GB

- Intel Xeon Phi 5110P

SMP узел:



Задержка доступа к памяти, тактов

	NODE
Кэш L1	4
Кэш L2	15
Кэш L3	40
DRAM	170
NUMA	400

Некоторые флаги оптимизации для компиляторов Intel

-O3

-ipo включает глобальную межпроцедурную оптимизацию

-static предотвращает линкование с общими библиотекам

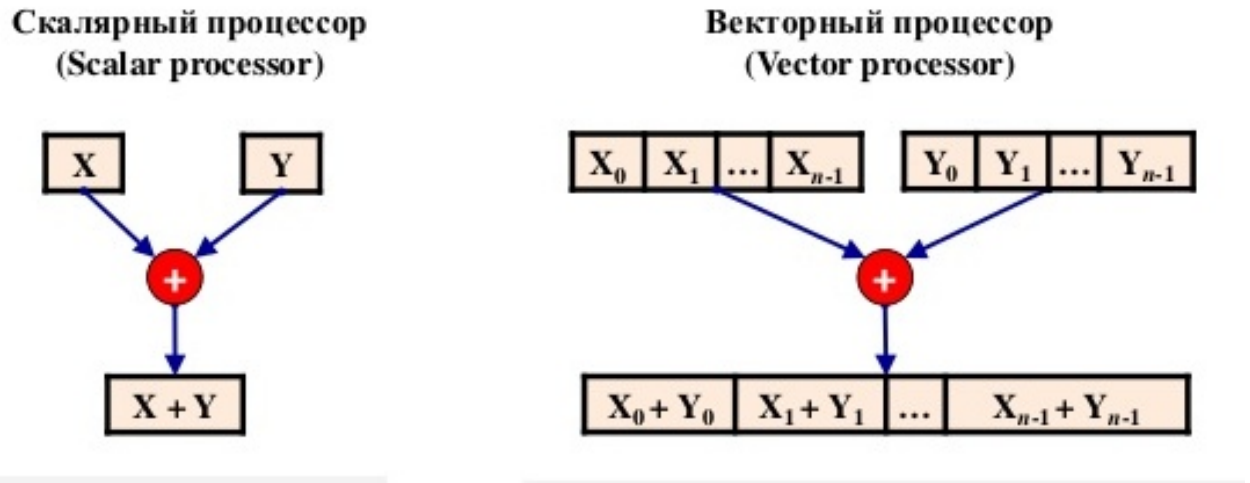
-xHost оптимизация (в том числе, векторизация) для того процессора, на котором запущен компилятор

-openmp включает поддержку OpenMP

-opt-report отчет об оптимизациях

Векторизация

Векторизация — оптимизация программы с использованием векторных расширений системы команд процессора



Базовые директивы

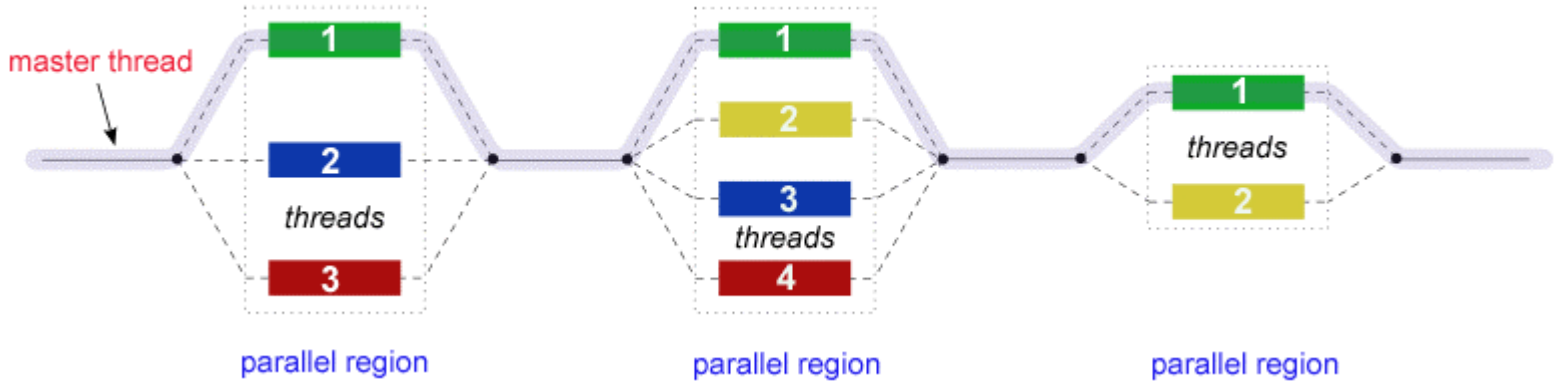
- ***#pragma ivdep*** - недоказанные зависимости в цикле не принимаются во внимание
- ***#pragma simd vectorlength(n1[, n2]…)*** - позволяет задать, какое количество итераций цикла будет одновременно исполняться при его векторизации. n должно быть степенью двойки.

Для проверки состояния векторизации удобно использовать ключ

–vec-report [0-5] – вывод отчета о векторизации (как правило, достаточно 3)

Технология OpenMP

OpenMP – технология программирования многопоточных приложений на многопроцессорных системах с общей памятью



Компилятор Intel icc 16.0.1 поддерживает стандарт OpenMP 4.0

Базовые директивы.

- ***#pragma omp parallel*** *[private(список переменных), num_threads(n)]* – начало параллельного региона

  private(список переменных) – список частных переменных для каждого потока

  num_threads(n) – задает количество создаваемых потоков

- ***#pragma omp for*** *[reduction(оператор: список переменных)]* - сообщает, что при выполнении цикла for в параллельном регионе итерации цикла должны быть распределены между потоками

  reduction(оператор: переменная) – обозначает переменную, с которой в цикле производится операция (например, +). При выходе из цикла, данная операция производится над копиями переменной во всех потоках, и результат присваивается оригинальной переменной.

32 подводных камня OpenMP при программировании на C++:
http://www.viva64.com/ru/a/0054

Переменные окружения OpenMP. Распределение системных ресурсов

- ***OMP_NUM_THREADS=n*** – задает количество создаваемых потоков
- ***KMP_AFFINITY="granularity=fine"*** – привязывает поток к аппаратному потоку
- ***KMP_AFFINITY="compact"***

  В этом случае на ядро будет приходиться максимально возможное число потоков, часть ядер будет свободна. Подходит приложениям с хорошей локальностью данных. Для остальных приложений может вызвать снижение производительности.
- ***KMP_AFFINITY="scatter"***

  Потоки равномерно распределяются по ядрам. Подходит для максимального использования системных ресурсов.
- ***KMP_AFFINITY=verbose*** – выводит информацию о текущем распределении ресурсов между потоками.
- ***KMP_STACKSIZE=16K*** – размер стека