

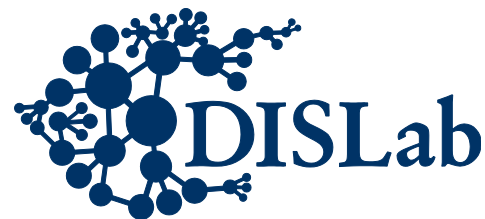
Спецкурс

«Параллельная обработка больших графов»

Лекция 2

к.т.н. Александр Сергеевич Семенов

dislab.org



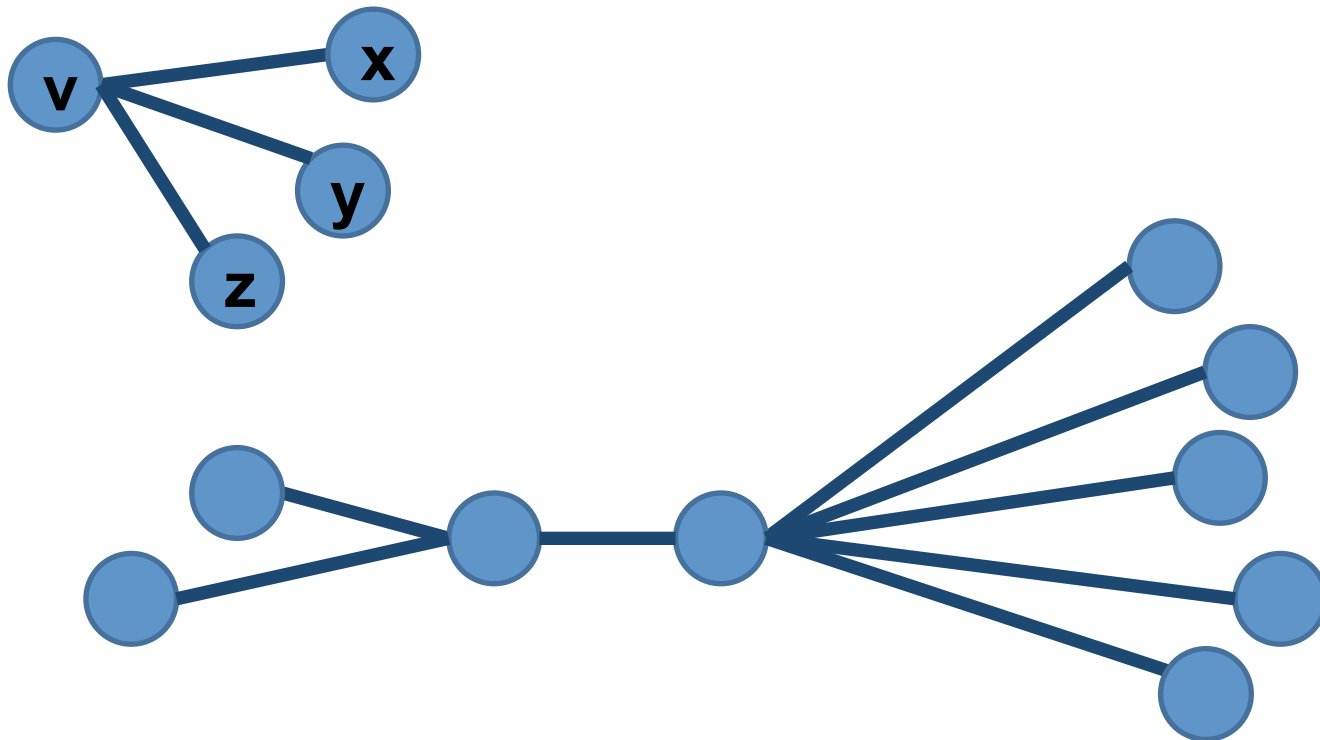
Основные проблемы, возникающие при решении задач обработки графов

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.

Проблемы анализа больших графов (1)

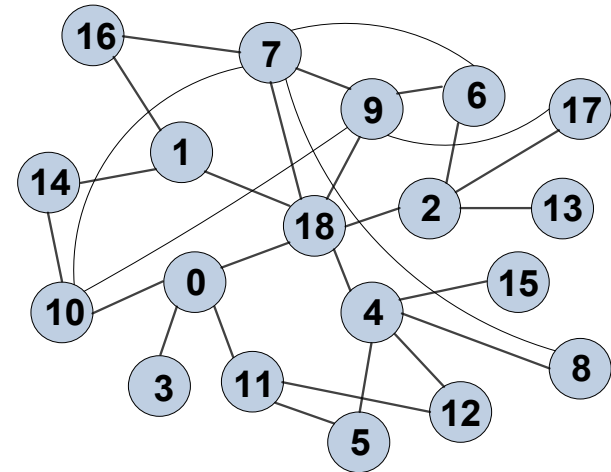
- **Data-driven computations.** Зависимость вычислений от данных (топологии графа). Невозможность применения методов статического распараллеливания вычислений.



Проблемы анализа больших графов (2)

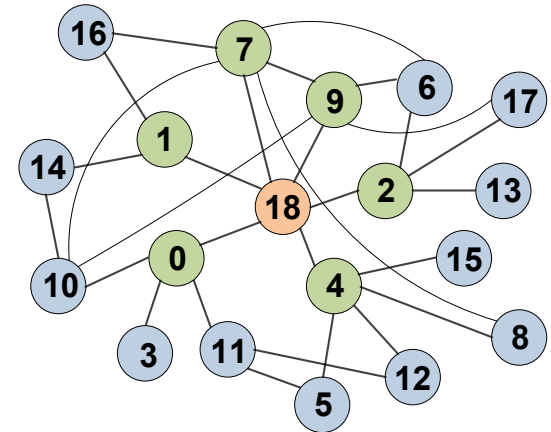
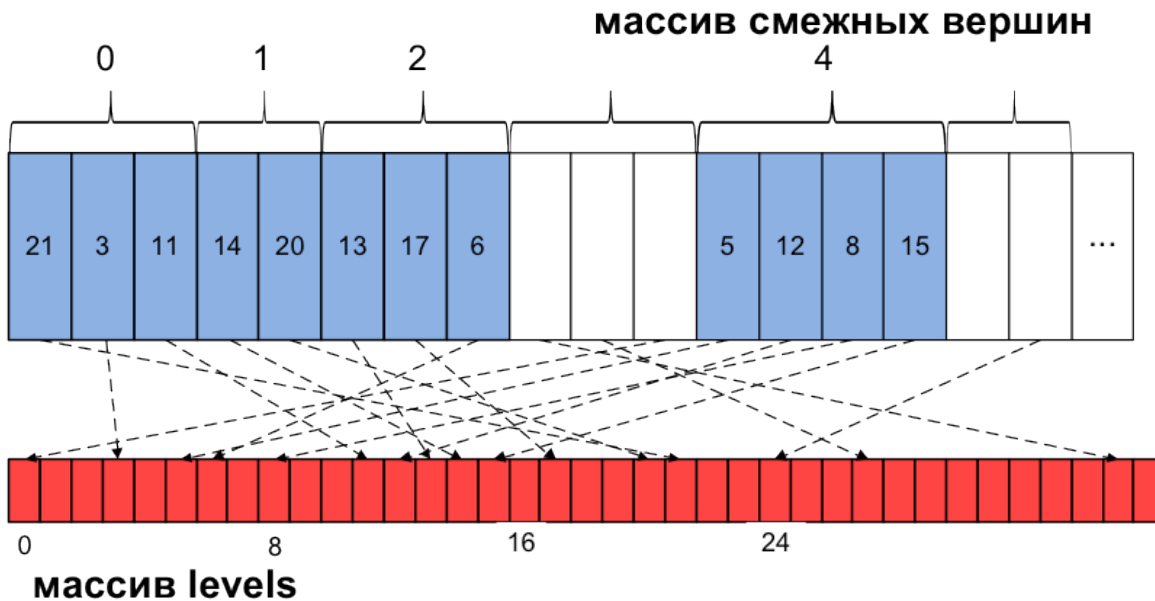
- **Unstructured problems.** Работа с нерегулярными, неструктурированными данными, трудность распараллеливания.

	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
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									



Проблемы анализа больших графов (3)

- **Poor locality.** Низкая пространственно-временная локализация обращений к памяти.



Проблемы анализа больших графов (4)

- **High data access to computation ratio.** Преобладание команд доступа к памяти над командами выполнения арифметических операций.

Intel E5-2680 v3, 2.5 ГГц

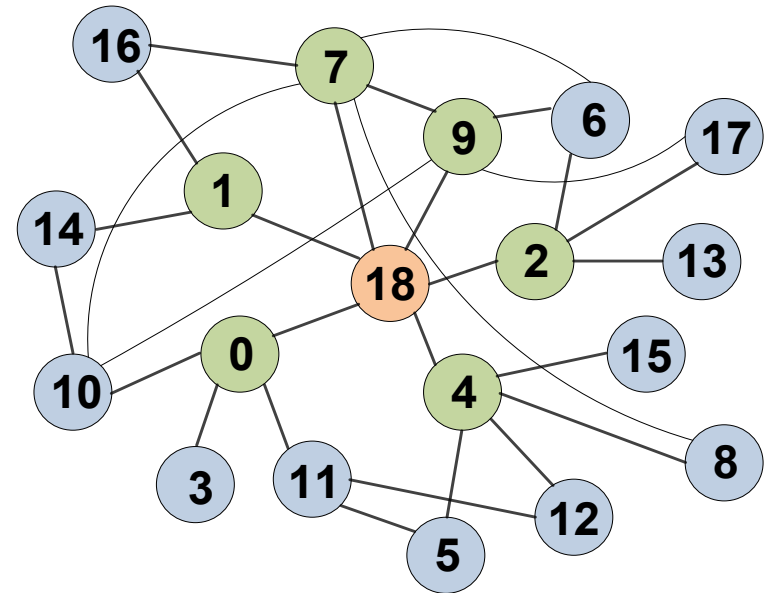
Параметр	Задержка, нс (такты)	ПС, ГБ/с
Регистр	(1)	–
Кэш L1	1.6 (4)	240
Кэш L2	4.4 (11)	160
Кэш L3	16 (40)	80
Память своего сокета	60	~55
Память чужого сокета	100	~30
Сеть Ангара	MPI – 1000 нс, SHMEM – 600 нс	

Алгоритмы обработки графов

Поиск вширь в графе (Breadth-first Search, BFS)

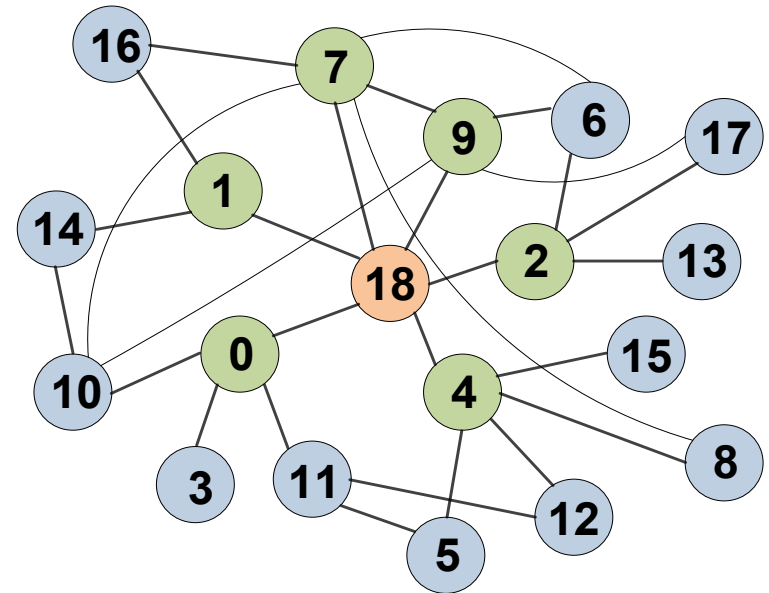
```
BFS ( $G, v_{start}$ )            $G = (V, E)$   
 $Q = \{v_{start}\}$   
 $Visited = \{v_{start}\}$   
while  $Q \neq \{\}$   
     $Q_{next} = \{\}$   
    for all vertex  $\in Q$  do  
        for all  $w: (vertex, w) \in E$  do  
            if  $w \notin Visited$  then  
                 $Q_{next} = Q_{next} \cup w$   
                 $Visited = Visited \cup w$   
            endif  
        end for  
    end for  
     $Q = Q_{next}$   
end while
```

СЛОЖНОСТЬ $O(N+M)$



Поиск вширь в графе (Breadth-first Search, BFS)

```
BFS ( $G, v_{\text{start}}$ )            $G = (V, E)$   
 $Q = \{v_{\text{start}}\}$   
 $\text{Visited} = \{v_{\text{start}}\}$   
while  $Q \neq \{\}$   
     $Q_{\text{next}} = \{\}$   
    #pragma omp parallel for  
    for all vertex  $\in Q$  do  
        for all  $w: (\text{vertex}, w) \in E$  do  
            if  $w \notin \text{Visited}$  then  
                #pragma omp critical  
                 $Q_{\text{next}} = Q_{\text{next}} \cup w$   
                 $\text{Visited} = \text{Visited} \cup w$   
            endif  
        end for  
    end for  
     $Q = Q_{\text{next}}$   
end while
```



Поиск вглубь в графе (Depth-first Search, DFS)

StartDFS (G, v_{start}) $G=(V, E)$

Visited = $\{v_{\text{start}}\}$

DFS(G, v_{start})

DFS (G, vertex)

Visited = Visited \cup vertex

for all $w: (\text{vertex}, w) \in E$ **do**

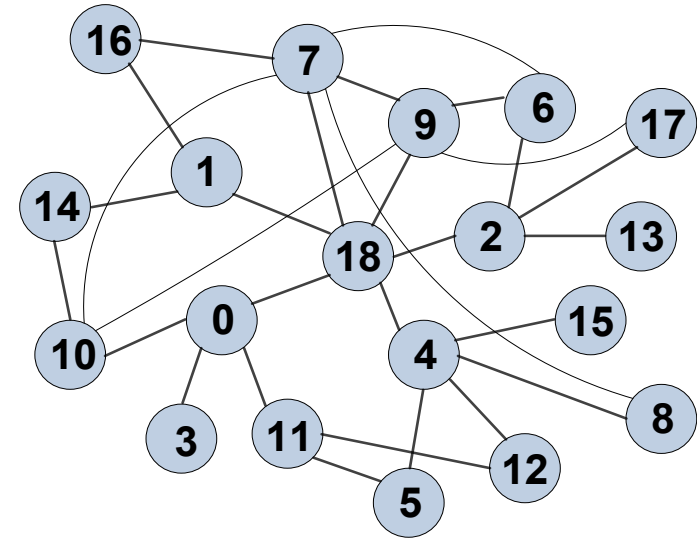
if $w \notin \text{Visited}$ **then**

 DFS(G, w)

endif

end for

СЛОЖНОСТЬ $O(N+M)$



Поиск всех кратчайших путей от заданной вершины в графе (Single Source Shortest Paths, SSSP)

$$G = (V, E, W), W : E \rightarrow R, \textit{undirected}$$

$$p(v_0, v_k) = \langle v_0, v_1, \dots, v_k \rangle$$

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$$

$$\delta(u, v) = \begin{cases} \min\{w(p)\}, \exists p(u, v) \\ \infty, \textit{otherwise} \end{cases}$$

SSSP, Вспомогательные процедуры

InitializeSingleSource(G, s)

for all $v \in V$

$d[v] = \infty$

end for

$d[s] = 0$

Relax(u, v, w)

if $d[v] > d[u] + w(u, v)$

$d[v] = d[u] + w(u, v)$

$p[v] = u$

end if

SSSP, Алгоритм Дейкстры

```
Dijkstra (G, w, s)
InitializeSingleSource(G, s)
S = {}
Q = V
while Q != {}
    u = GetMin (Q)
    S = S U {u}
    Q = Q / {u}
    for all v in Adj[u]
        Relax (u, v, w)
    end for
end while
```

- $G(V, E, W)$ – ориентированный, $w(e) \geq 0$, $w \in W$
- последовательный

Сложность $O(N^2+M)$, $O(N \lg N + M)$

SSSP, Алгоритм Беллмана-Форда

```
BellmanFord (G, s)
InitializeSingleSource(G, s)
for i = 1 to |V| - 1
    for all (u, v) in E
        Relax(u, v, w)
    end for
end for
```

- $G(V, E, W)$ – ориентированный, $W: E \rightarrow \mathbb{R}$
- может определять наличие циклов с отрицательным весом

Сложность $O(N M)$

SSSP, Алгоритм Беллмана-Форда

BellmanFord (G, s)

InitializeSingleSource(G, s)

for i = 1 to |V| - 1

#pragma omp parallel for

for all (u, v) in E

Relax(u, v, w) // **атомарно**

end for

end for

SSSP, Алгоритм delta-stepping (1/4)

- $G(V, E, W)$ – ориентированный, $w(e) \geq 0$, w in W
- δ in R , $\delta \geq 0$

DeltaStepping (G, w, s, δ)

InitializeDeltaStepping

$i = 0$

while true

 processBucket(i)

$i = \min(k > i : B_k \neq \{\})$

if $i \neq \text{inf}$ **then break**

end while

SSSP, Алгоритм delta-stepping (2/4)

InitializeDeltaStepping (G, w, s)

for all u in V

 d(u) = inf

end for

d(s) = 0

$B_0 = \{s\}$

$B_{\text{inf}} = V \setminus \{s\}$

$B_1 = B_2 = \dots = \{\}$

$$B_i = \{v \in V : d[v] \in [i * \text{delta}; (i + 1) * \text{delta}]\}$$

SSSP, Алгоритм delta-stepping (3/4)

ProcessBucket (i)

$A = B_i$

while $A \neq \{\}$

for all u **in** A , **for all** $e = (u, v)$ **in** E

 Relax(u, v)

end for

$A' = \{x : d(x) \text{ изменен в предыдущем цикле}\}$

$A = B_i \cap A'$

end while

SSSP, Алгоритм delta-stepping (4/4)

Relax (u, v)

$i = \lfloor d(v) / \text{delta} \rfloor$ // старый bucket

$d(v) = \min(d(v), d(u) + w(u, v))$

$j = \lfloor d(v) / \text{delta} \rfloor$ // новый bucket

if $j < i$ **then** переместить v из B_i в B_j

SSSP, Алгоритм delta-stepping

DeltaStepping (G, w, s, delta)

InitializeDeltaStepping

i = 0

while true

 processBucket(i)

 i = min(k > i : B_k != {})

if i != inf **then break**

end while

- Для случайных графов «хороший» показатель $\delta = 1/d$, где d – максимальная степень вершины, w in [0; 1]

- Сложность $O(d N)$

- Количество итераций

- d_c - максимальная длина пути

$$O\left(\frac{d_c}{\delta} * \frac{\log n}{\log \log n}\right)$$

SSSP, Алгоритм delta-stepping

ProcessBucket (i)

$A = B_i$

while $A \neq \{\}$

#pragma omp parallel for

for all u in A , **for all** $e = (u, v)$ in E

Relax(u, v) // **атомарно** _

end for

$A' = \{x : d(x) \text{ changed in previous for all}\}$

$A = B_i \cap A'$

end while