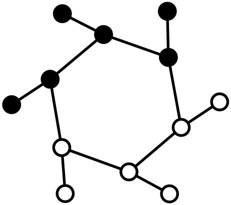
Advances in Databases and Information Systems
Genoa, September 4, 2013



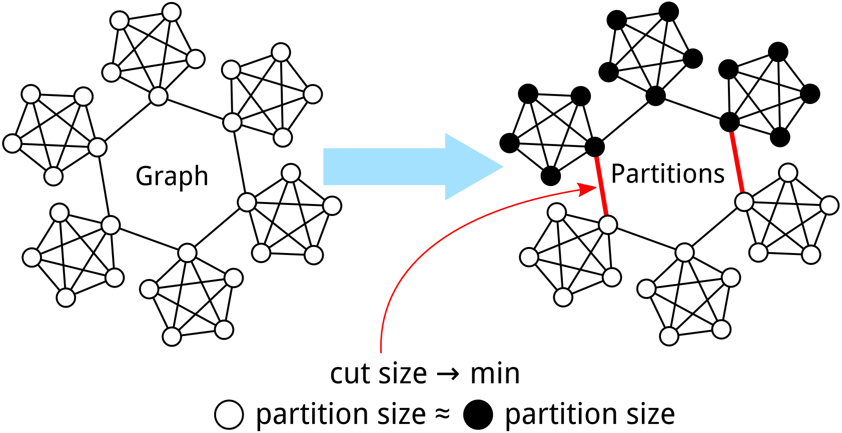
Very Large Graph Partitioning by Means of Parallel DBMS

Constantin Pan, Mikhail Zymbler

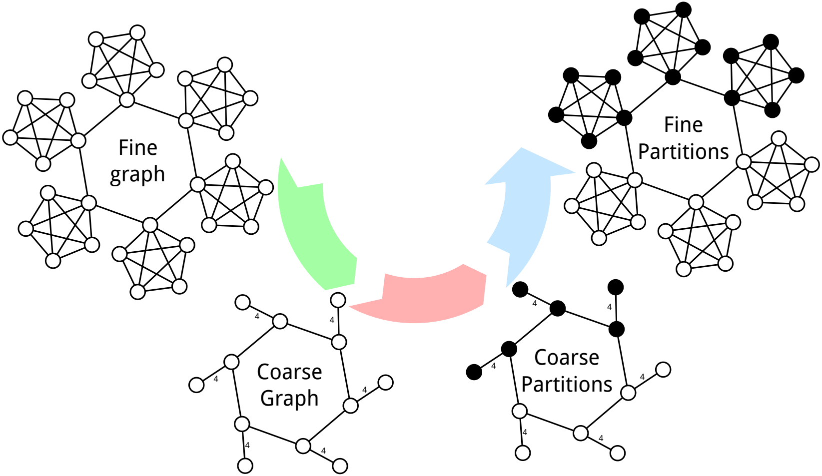
South University,
Chelyabinsk, Russia

The reported study was partially supported by the Russian Foundation for Basic Research, research projects No. 12-07-31217 and No. 12-07-00443.

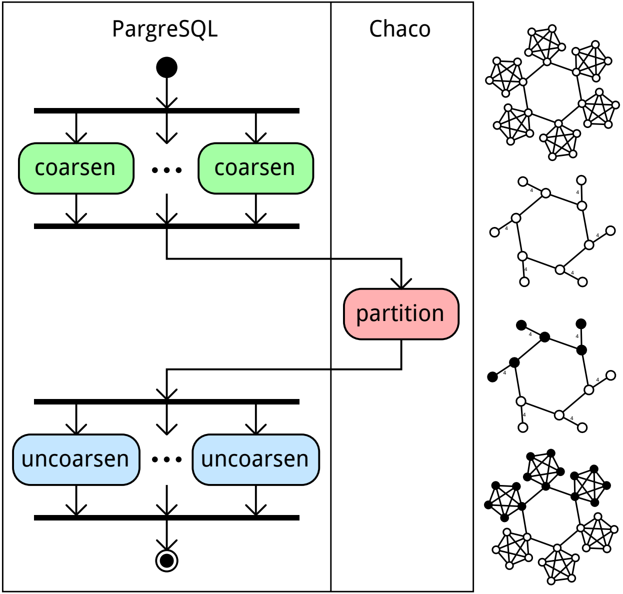
Graph Partitioning



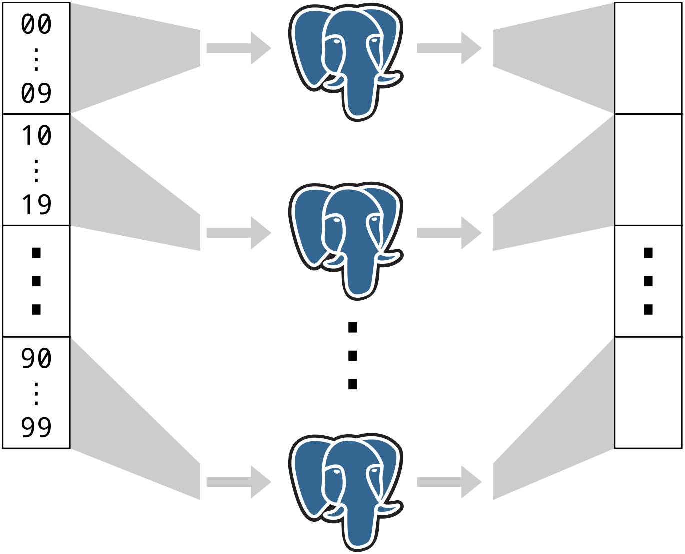
Multilevel Partitioning



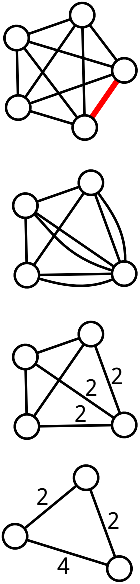
Using Parallel DBMS



ParGreSQL

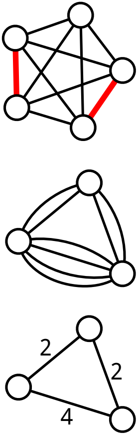


Coarsening



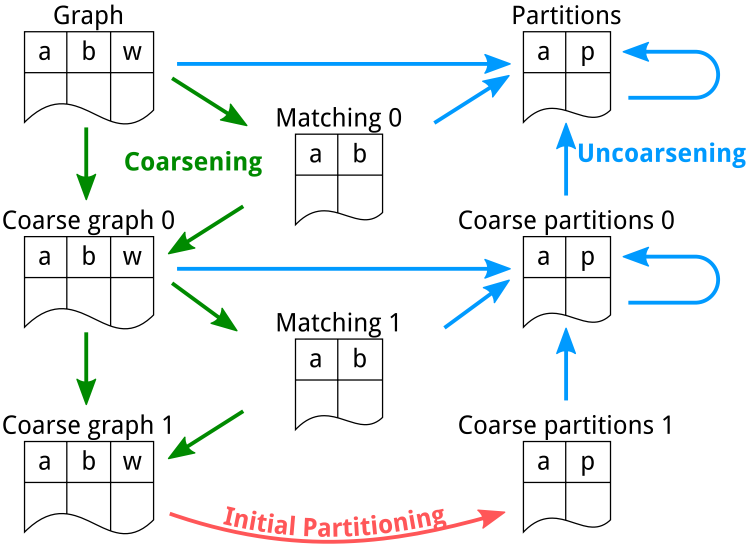
1. Find the heaviest (or a random) edge.
2. Collapse the edge into a vertex.
3. Merge the duplicates and remove the loops.
4. Repeat, avoiding the vertices generated this way, until nothing is left.

Coarsening with DBMS



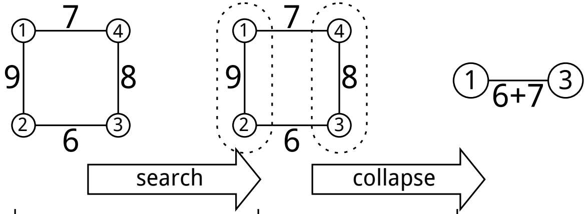
1. Find the heaviest matching.
2. Collapse the edges of the matching into vertices.
3. Merge the duplicates and remove the loops.

Data Flow



Graph
a
b
w
Matching 0
a
b
Coarsening
Partitions
a
p
Uncoarsening
Coarse graph 0
a
b
w
Coarse partitions 0
a
p
Matching 1
a
b
Coarse graph 1
a
b
w
Coarse partitions 1
a
p
Initial Partitioning

Coarsening Implementation



GRAPH

A	B	W
1	2	9
2	3	6
3	4	8
4	1	7

MATCH

A	B
1	2
3	4

COARSE GRAPH

A	B	W
1	3	13

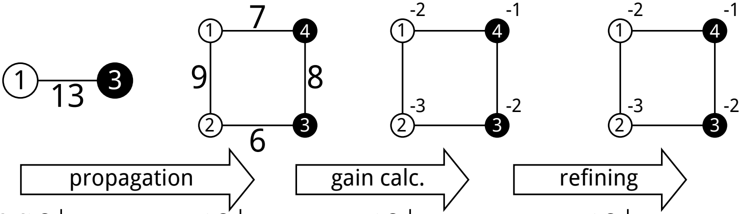
Coarsening Implementation

```
-- search

for edge in (select A,B from GRAPH order by W desc)
loop
  if not exists(
    select * from visited where A = edge.A or A = edge.B
  ) then
    insert into visited values (edge.A);
    insert into visited values (edge.B);
    insert into MATCH values (edge.A, edge.B);
  end if;
end loop;
```

Coarsening Implementation

```
-- collapse

select
  least(newA, newB) as A,
  greatest(newA, newB) as B,
  sum(W) as W
from (
  select
    coalesce(match2.A, GRAPH.A) as newA,
    coalesce(MATCH.A, GRAPH.B) as newB,
    GRAPH.W
  from
    GRAPH, left join MATCH on GRAPH.B=MATCH.B
    left join MATCH as match2 on GRAPH.A=match2.B)
where newA != newB group by A, B;
```

Uncoarsening Implementation



COARSE PARTITIONS

A	P
1	1
3	0

PARTITIONS

A	P
1	1
2	1
3	0
4	0

PARTITIONS

A	P	G
1	1	-2
2	1	-3
3	0	-2
4	0	-1

PARTITIONS

A	P	G
1	1	-2
2	1	-3
3	0	-2
4	0	-1

Coarsening Implementation

```
-- propagate

select a, p from COARSE_PARTS
union
select match.b, part.p
from MATCH as match, COARSE_PARTS as part
where match.a = part.a;
```

Coarsening Implementation

```
-- calculate gains

select
  PARTITIONS.A, PARTITIONS.P,
  sum(subgains.Gain) as Gain
from
  PARTITIONS left join (
    select GRAPH.A, GRAPH.B,
      case when ap.P = bp.P then -GRAPH.W
        else GRAPH.W end as Gain
    from
      GRAPH left join PARTITIONS as ap on GRAPH.a = ap.A
      left join PARTITIONS as bp on GRAPH.b = bp.A
  ) as subgains
    on PARTITIONS.A = subgains.A
    or PARTITIONS.A = subgains.B
group by PARTITIONS.A, PARTITIONS.P;
```

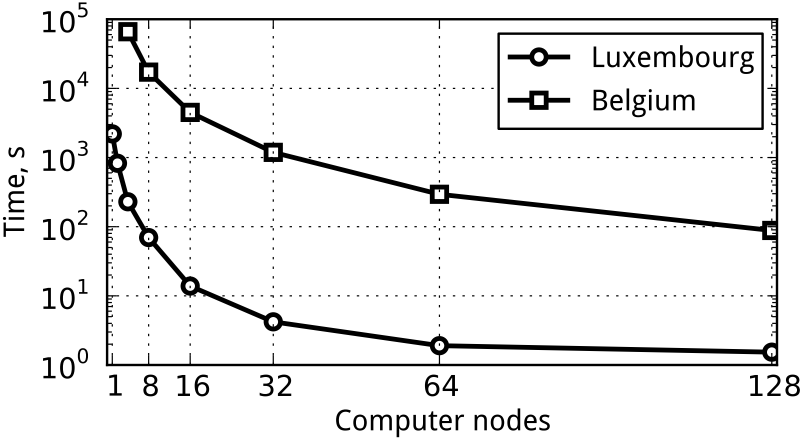
Coarsening Implementation

```
-- refine

select * from PARTITIONS
where P = current and G = (select max(G) from PARTITIONS
    where P = current)
limit 1 into V;

update PARTITIONS
  set G = G + W * (case when P = V.P then 2 else -2 end)
from (
  select case when A = V.A then B else A end, W from GRAPH
  where B = V.A or A = V.A) as neighbors
where neighbors.A = PARTITIONS.A;

update PARTITIONS
  set G = -G, P = 1 - P
where A = V.A;
```

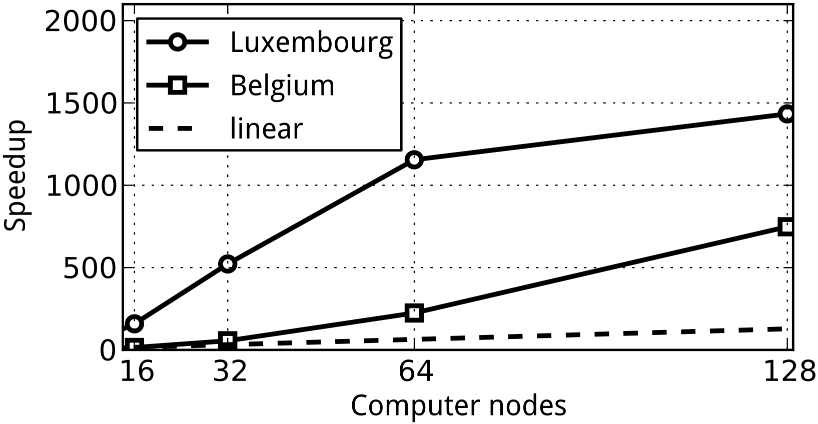
Experiments

- Computer
  - 128 nodes of Tornado cluster in South Ural State University (471st in top500)
- Data
  - Luxembourg road map from OpenStreetMap (10^5 vertices, 1 iteration)
  - Belgium road map from OpenStreetMap (10^6 vertices, 5 iterations)
  - distributed over the cluster nodes
    by function $\varphi(e) = e.A * |V|/|E|$

Execution time

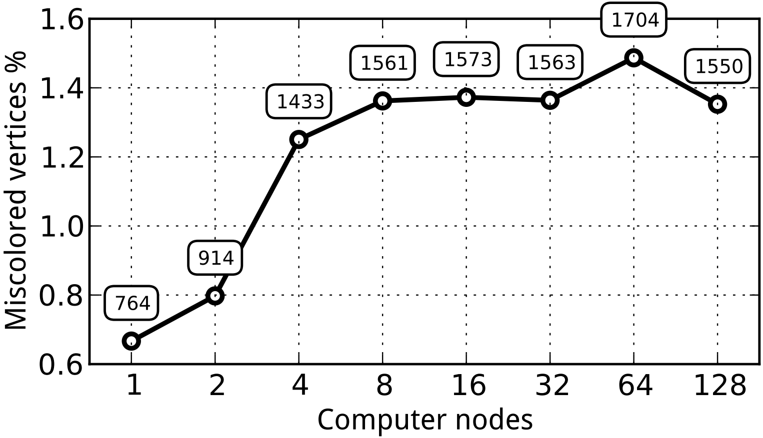


Speedup



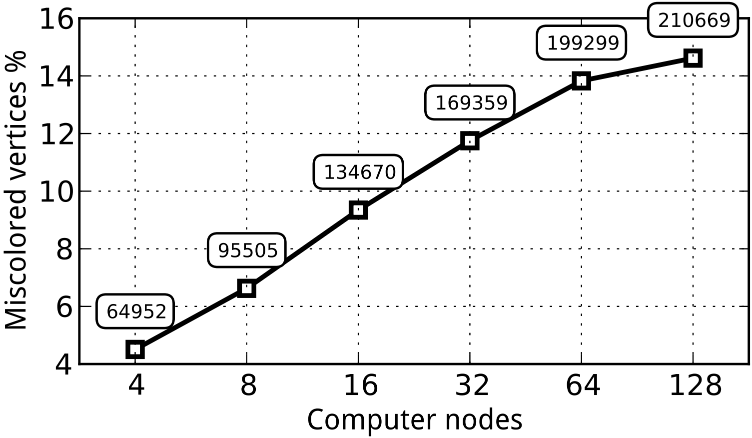
2000
1500
1000
500
0
Speedup
Luxembourg
Belgium
linear
16
32
64
128
Computer nodes

Quality (Luxembourg)



Random partitioning gives 30 % miscolored vertices.

Quality (Belgium)



Random partitioning gives 30 % miscolored vertices.

Conclusions

- A new approach to partition very large graphs by means of a relational parallel DBMS, that was implemented on the basis of PostgreSQL.
- Good speedup at an acceptable quality loss.
- Try different partitioning schemes and other very large graph problems in future.

Thank you

Questions?

Constantin Pan
kvapen@gmail.com

Mikhail Zymbler
zymbler@gmail.com