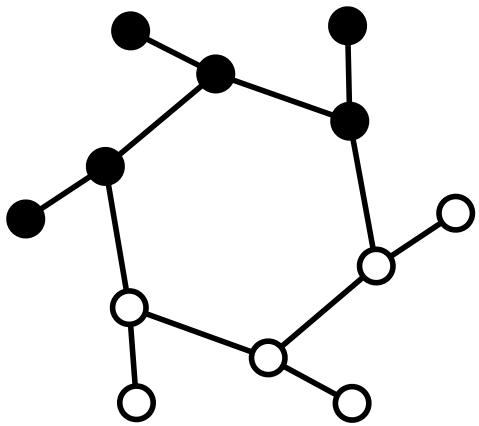
Advances in Databases and Information Systems
Genoa, September 4, 2013



Very Large Graph Partitioning by Means of Parallel DBMS

Constantin Pan, Mikhail Zymbler

South Ural State 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.

2013-09-04

Very Large GraphPartitioningby Means ofParallel DBMS

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

Very Large Graph Partitioning by Means of Parallel DBMS

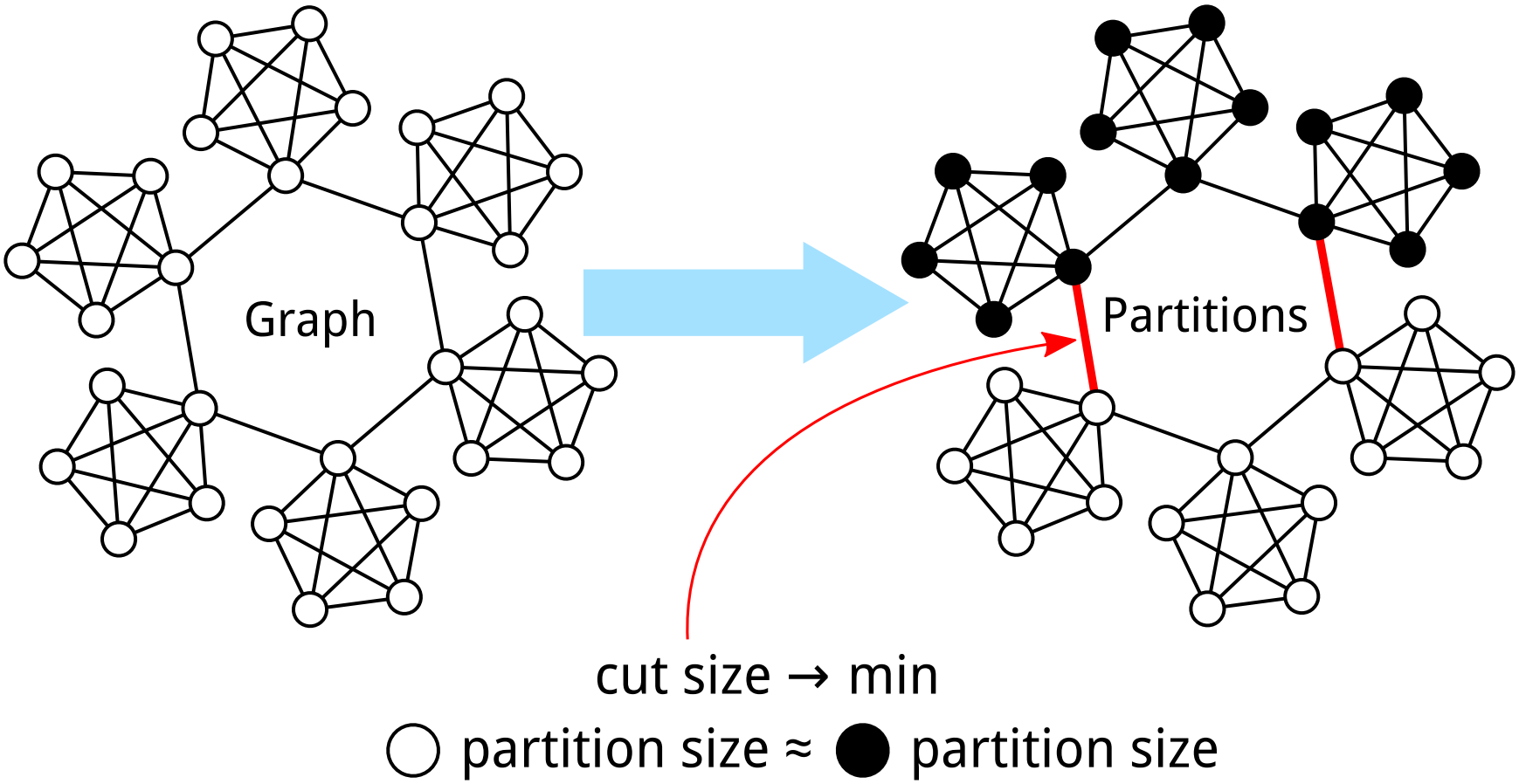
Constantin Pan, Mikhail Zymbler

South Ural State 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.

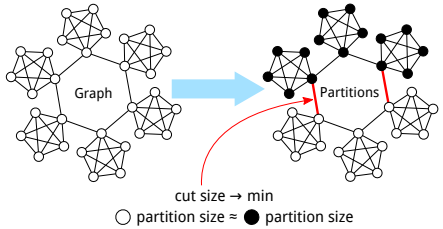
Good day, everyone! I will present a new approach to very large graph partitioning that involves a parallel database management system.

Graph Partitioning



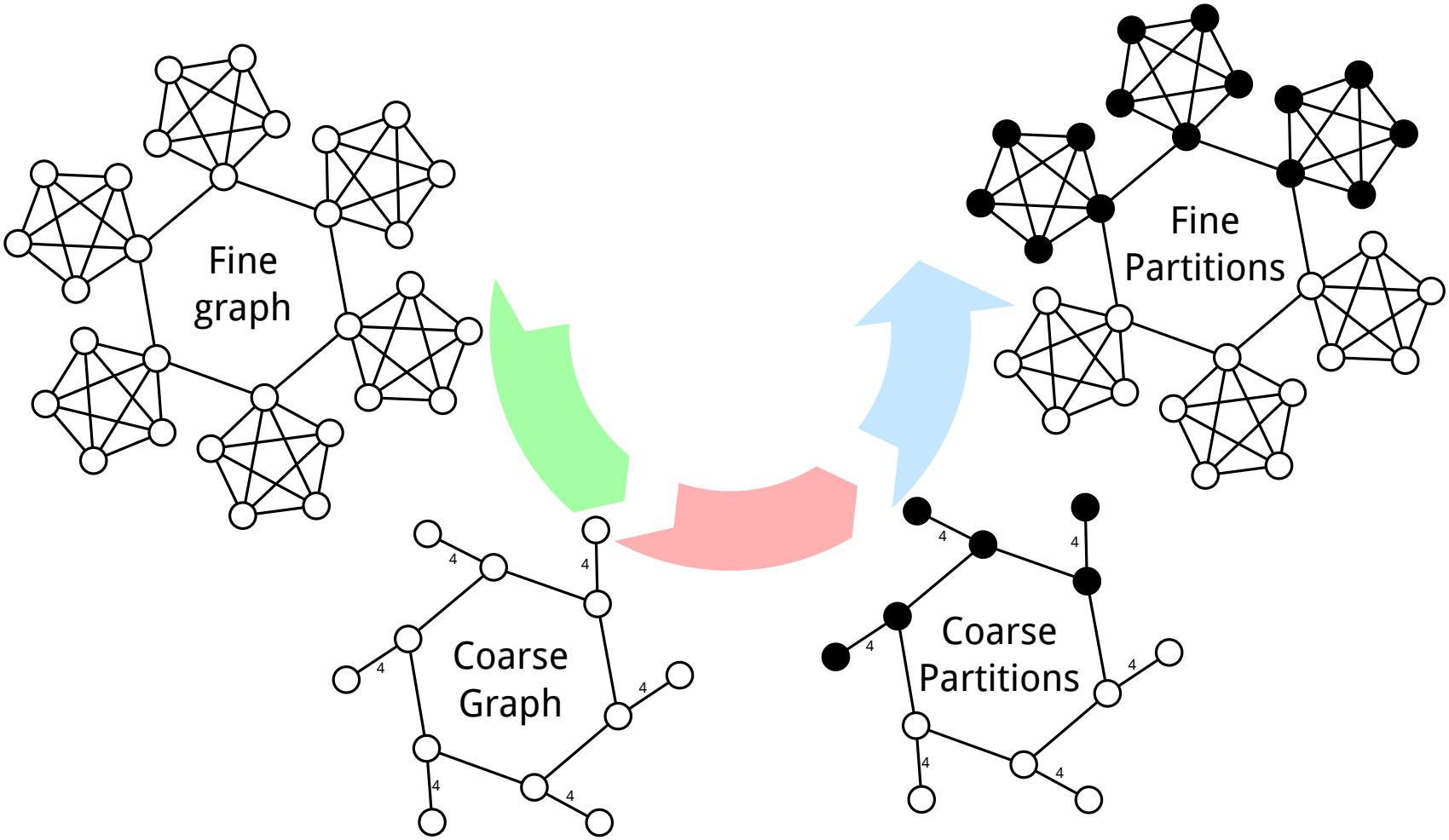
Very Large GraphPartitioningby Means ofParallel DBMS

Graph Partitioning



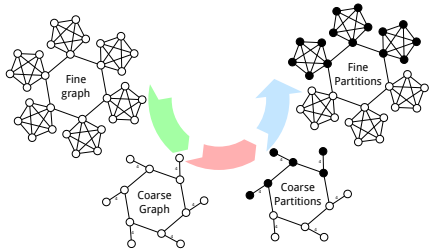
The graph partitioning problem is about cutting a graph into partitions so that the cut size would be minimal and the partitions would be of the same size. Since the multi-way partitioning can be achieved with recursive bisection we concentrated our research on the bisection problem.

Multilevel Partitioning



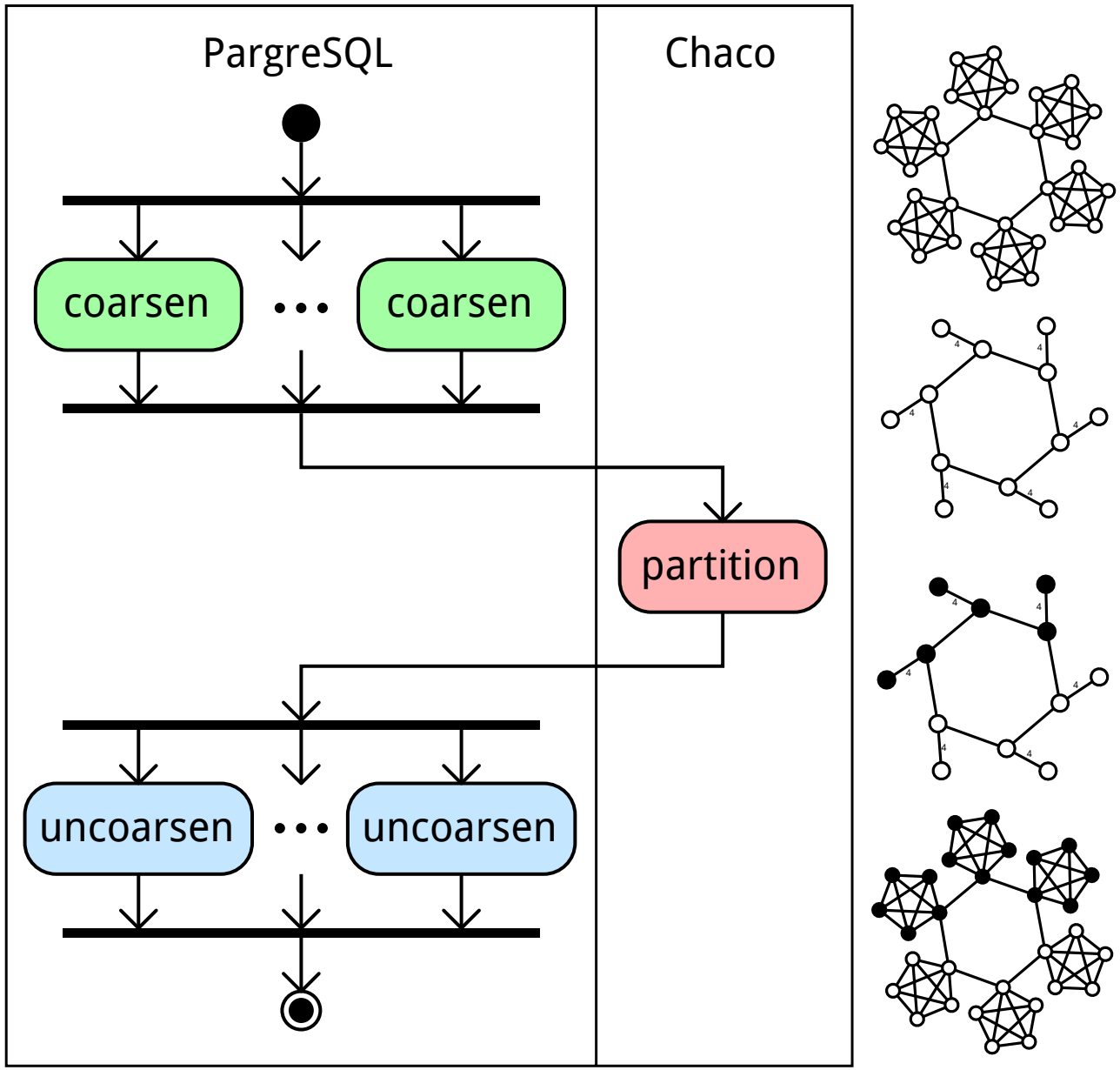
Very Large GraphPartitioningby Means ofParallel DBMS

Multilevel Partitioning



The usual way of partitioning very large graphs is the multilevel partitioning. In multilevel partitioning we first coarsen the graph until it becomes of a managable size, then apply some simple partitioning technique, and finally uncoarsen the results.

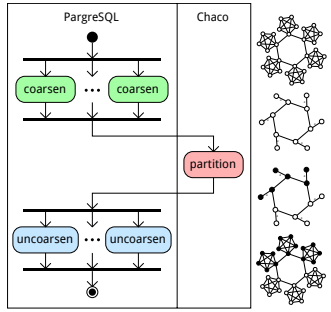
Using Parallel DBMS



Very Large GraphPartitioningby Means ofParallel DBMS

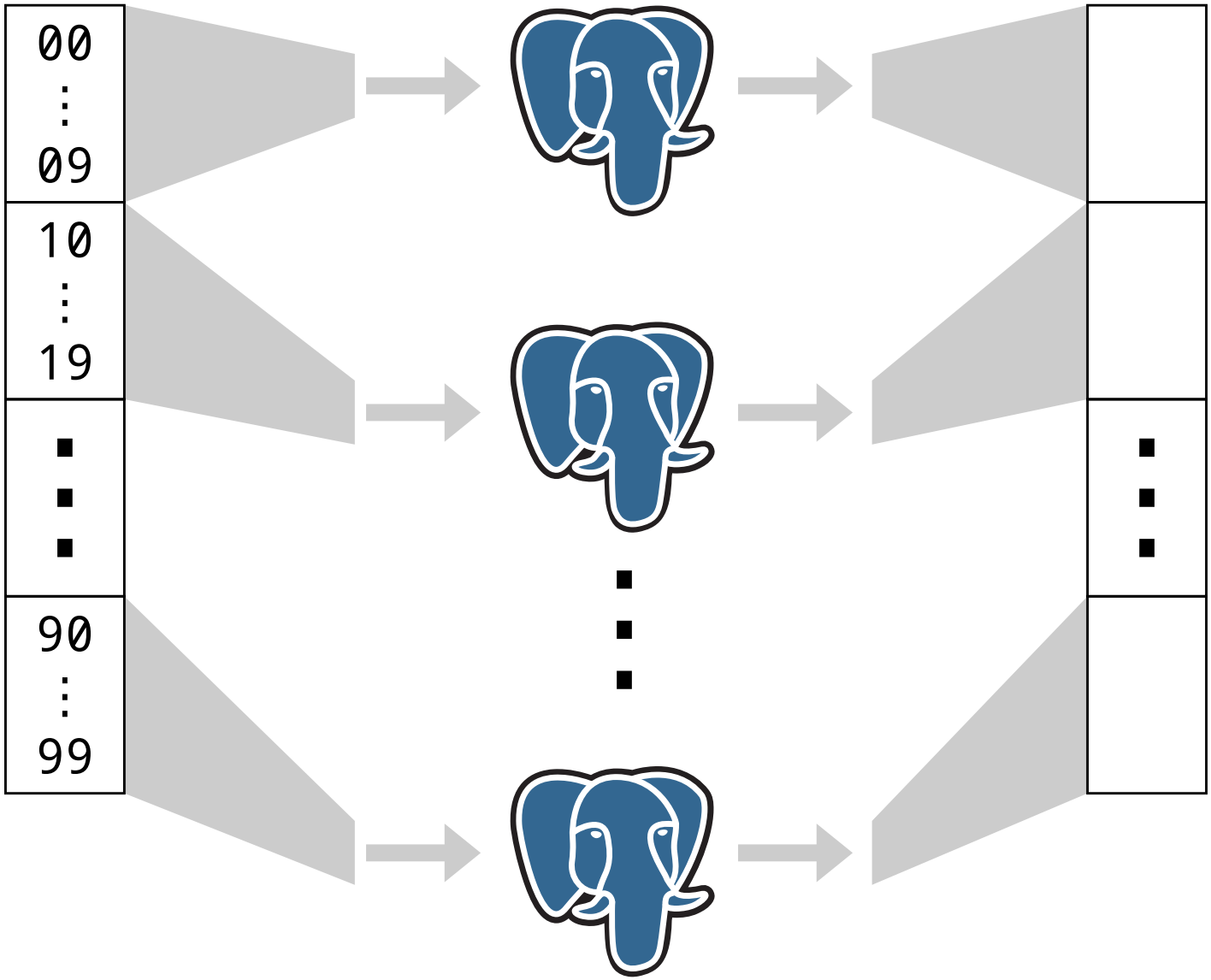
Using Parallel DBMS

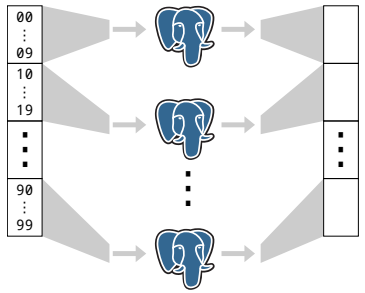
Using Parallel DBMS



We apply a parallel DBMS to do the coarsening and uncoarsening, and a third-party tool (Chaco) to do the initial partitioning of the coarse graph.

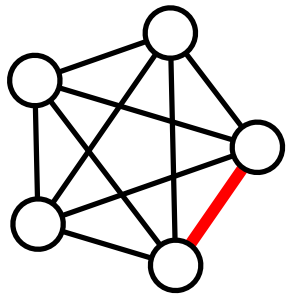
PargreSQL

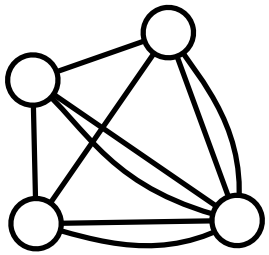


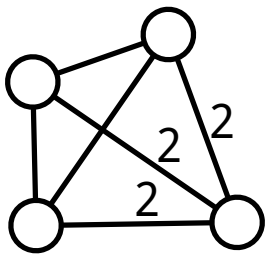


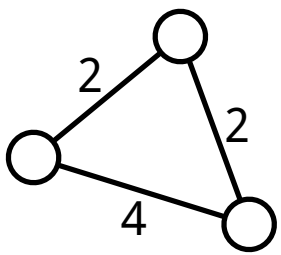
The parallel DBMS that we use is another project of our university. It is called PargreSQL since it is a parallel version of PostgreSQL that uses horizontal distribution of tables to implement parallelism.

Coarsening







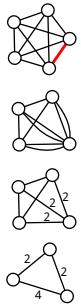


1. Find the heaviest (or a random) edge.
2. Collapse the edge into a vertex.
3. Merge the duplicates and remove the loops.
4. Repeat, avoiding the vertices generated this way, until nothing is left.

Very Large GraphPartitioningby Means ofParallel DBMS

Coarsening

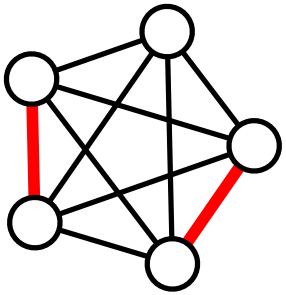
Coarsening

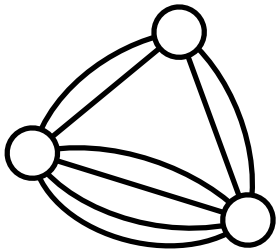


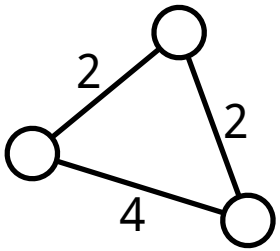
1. Find the heaviest (or a random) edge.
2. Collapse the edge into a vertex.
3. Merge the duplicates and remove the loops.
4. Repeat, avoiding the vertices generated this way, until nothing is left.

The basic method for graph coarsening is this...

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.

Very Large GraphPartitioningby Means ofParallel DBMS

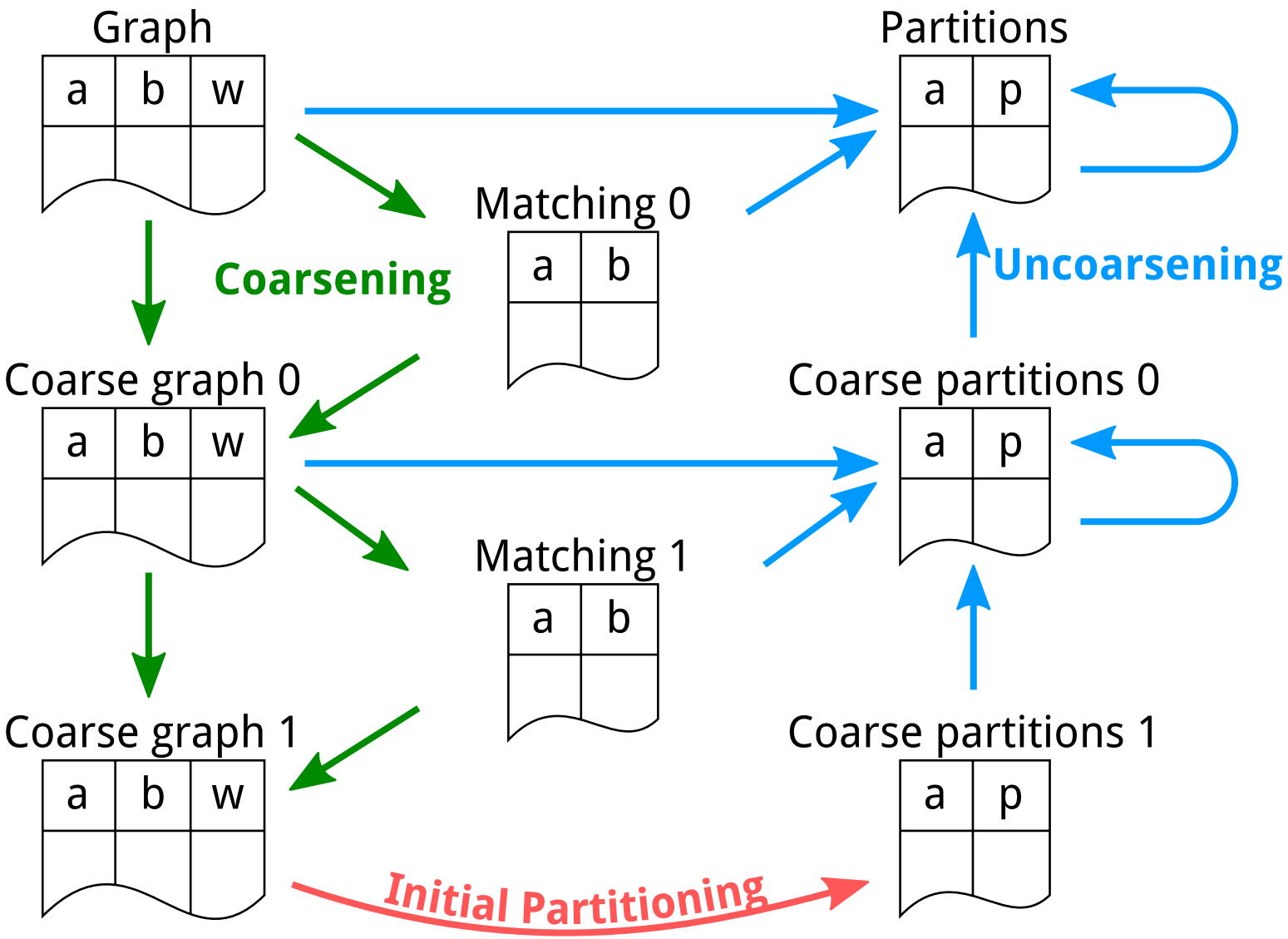
Coarsening with DBMS

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.

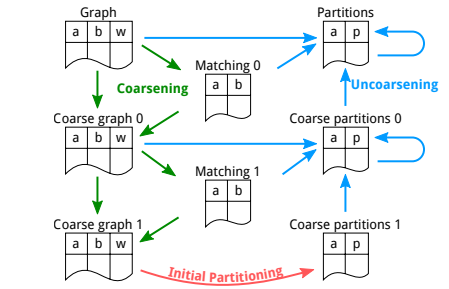
We made some changes to the coarsening method so that it would work better with the DBMS. In out method we do not collapse individual edges, but the whole independent edge set. This leads to the same result but with less queries.

Data Flow


Graph
a
b
w
Partitions
a
p
Matching 0
a
b
Coarsening
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

Very Large GraphPartitioningby Means ofParallel DBMS

Data Flow

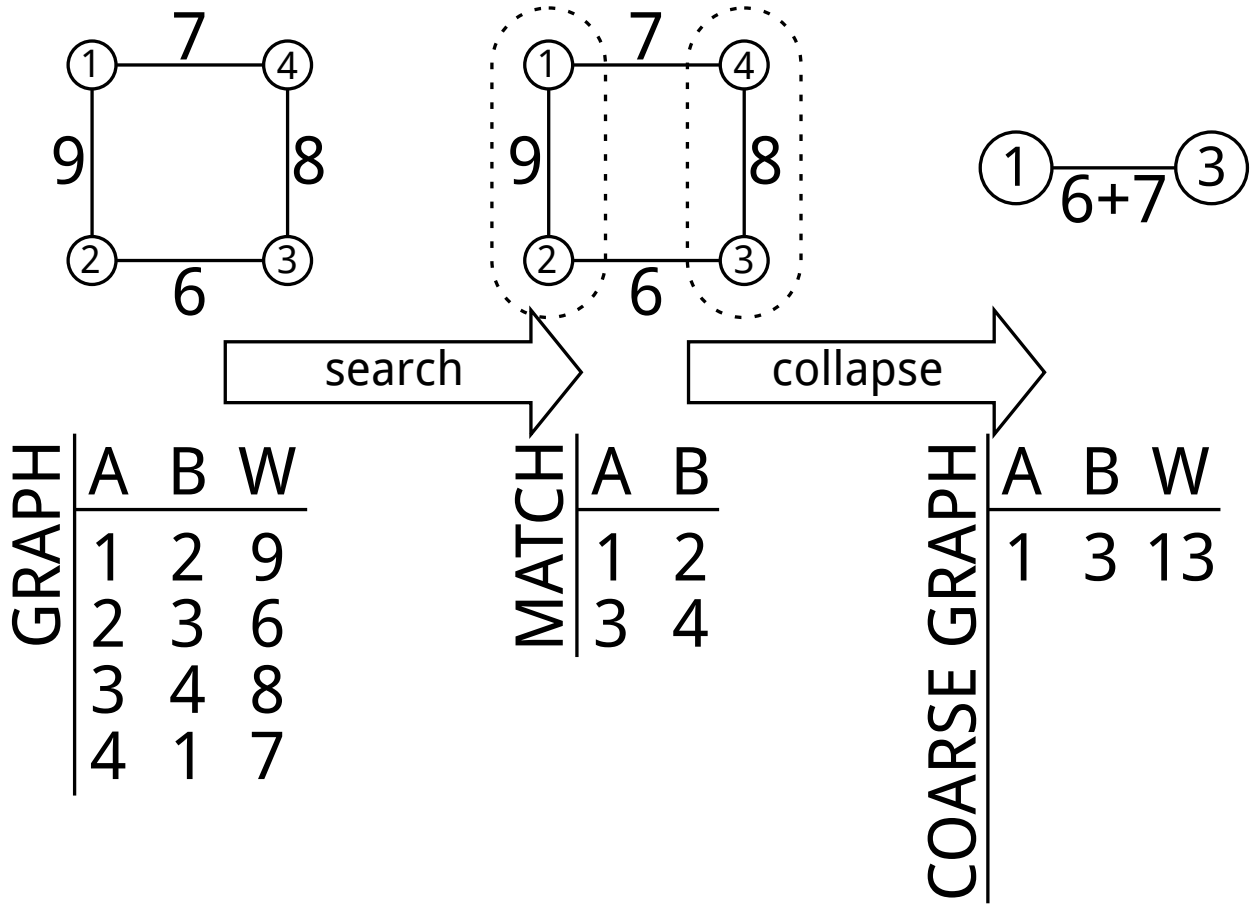


This slide shows the data flow during the multilevel partitioning process. The graphs are represented by lists of edges with weights. The matchings are just lists of edges. And partitions are lists of vertices with their partition number.

First we find the heaviest matching for the graph, then we collapse all the edges that are in this matching, and get the coarse graph. We can repeat the process as many times as needed, each time getting an even smaller graph.

The initial partitioning tool gives us a list of vertices with their corresponding partition numbers. We use the matching to propagate the partitioning values onto the vertices that were originally separate. Then we use the fine graph info to fix the errors introduced by this trivial propagation.

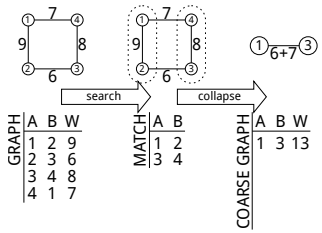
Coarsening Implementation



Very Large GraphPartitioningby Means ofParallel DBMS

Coarsening Implementation

Coarsening Implementation



This is an example of a graph getting coarsened. It shows the graph and the corresponding table values.

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;
```

Very Large GraphPartitioningby Means ofParallel DBMS

Coarsening Implementation

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;
```

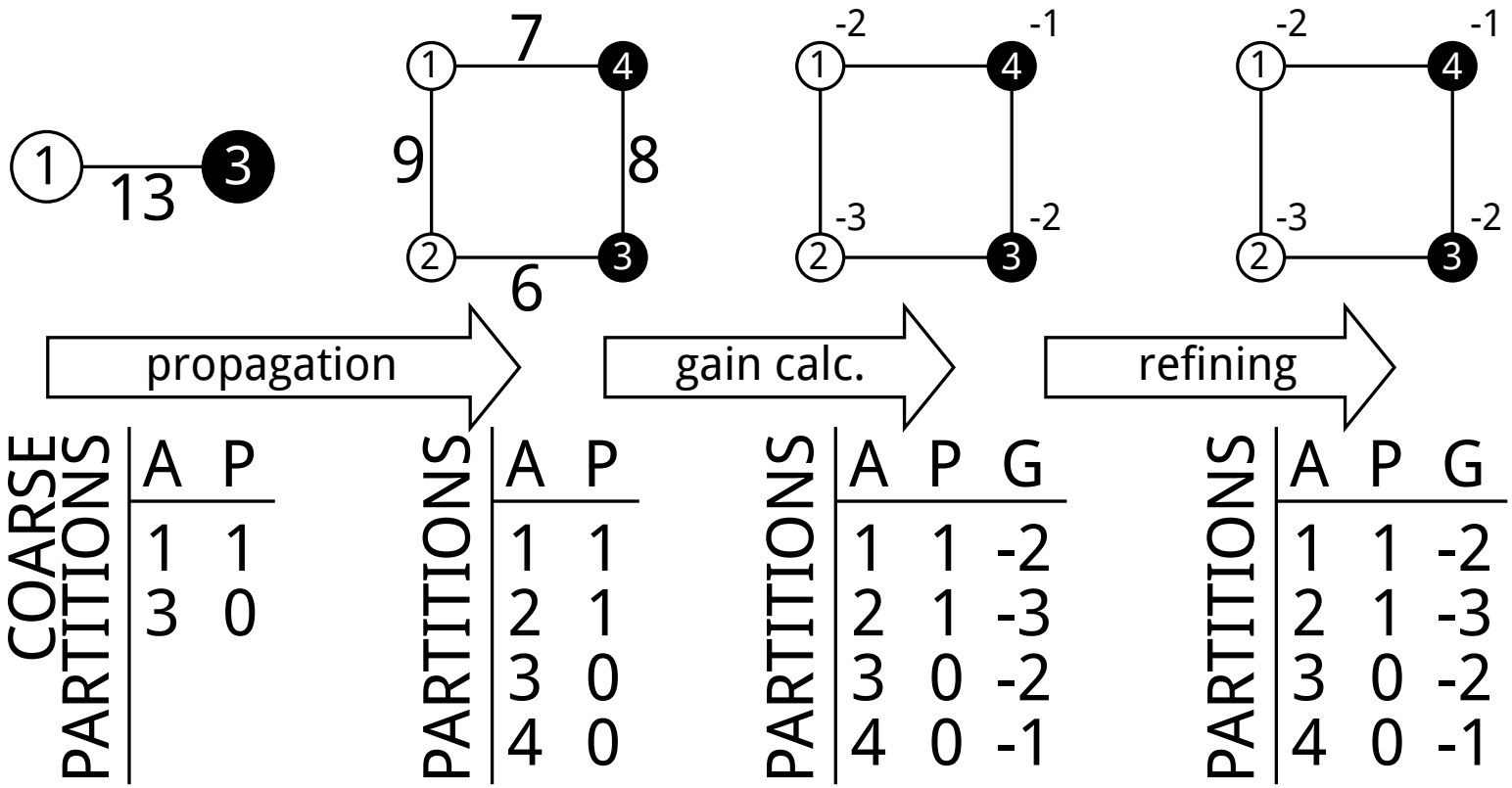
All the steps of coarsening are implemented in pgSQL and look like this...

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;
```

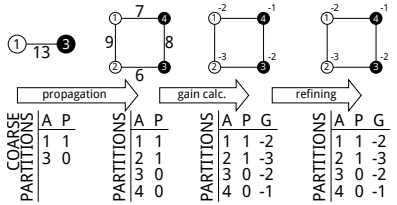
Uncoarsening Implementation



Very Large GraphPartitioningby Means ofParallel DBMS

Uncoarsening Implementation

Uncoarsening Implementation



This is an example of uncoarsening. It shows the two partitions with black and white color.

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 $\varphi(e) = e.A * |V|/|E|$

Very Large GraphPartitioningby Means ofParallel DBMS

Experiments

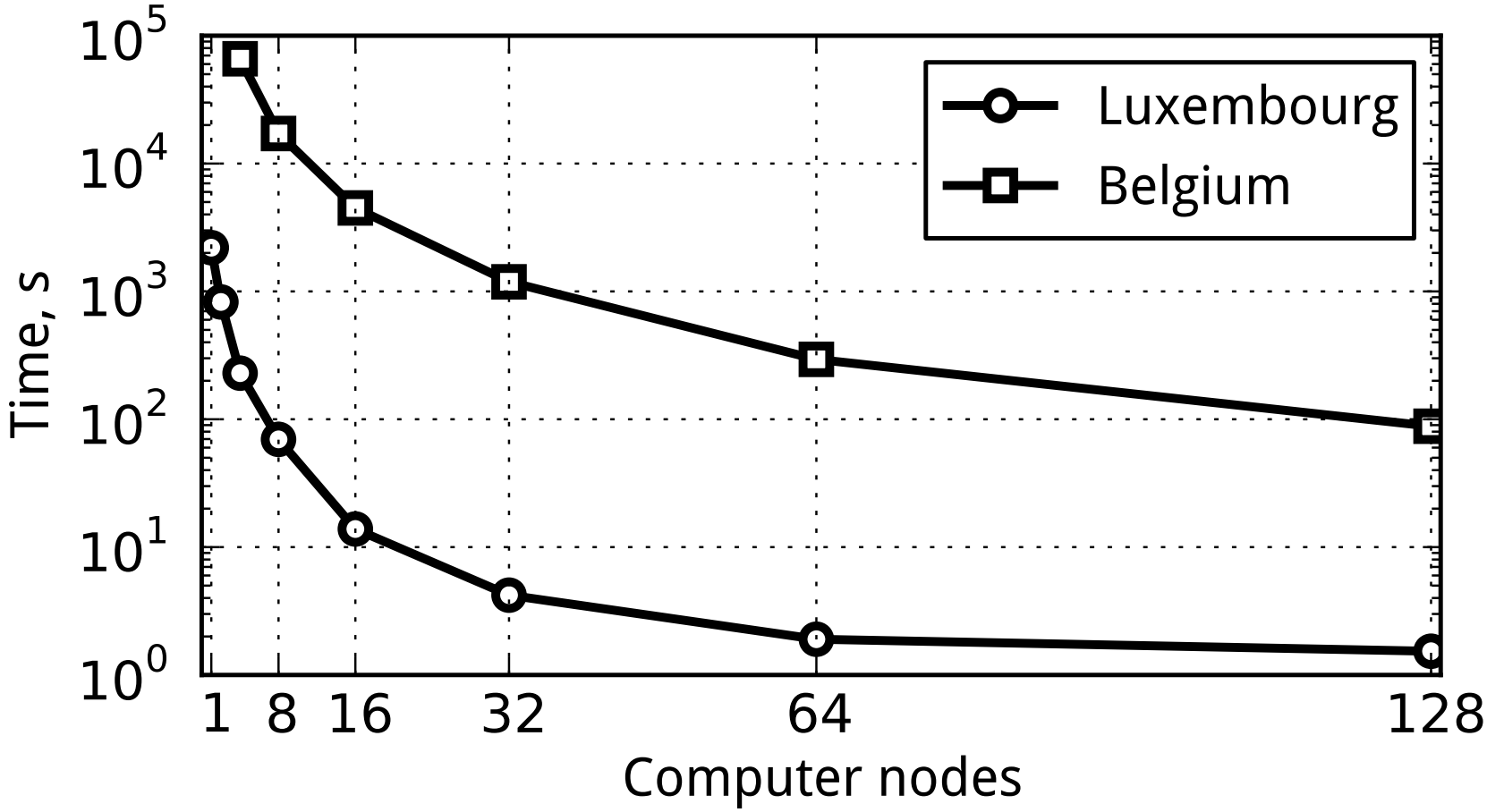
Experiments

- 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 $\varphi(e) = e.A * |V|/|E|$

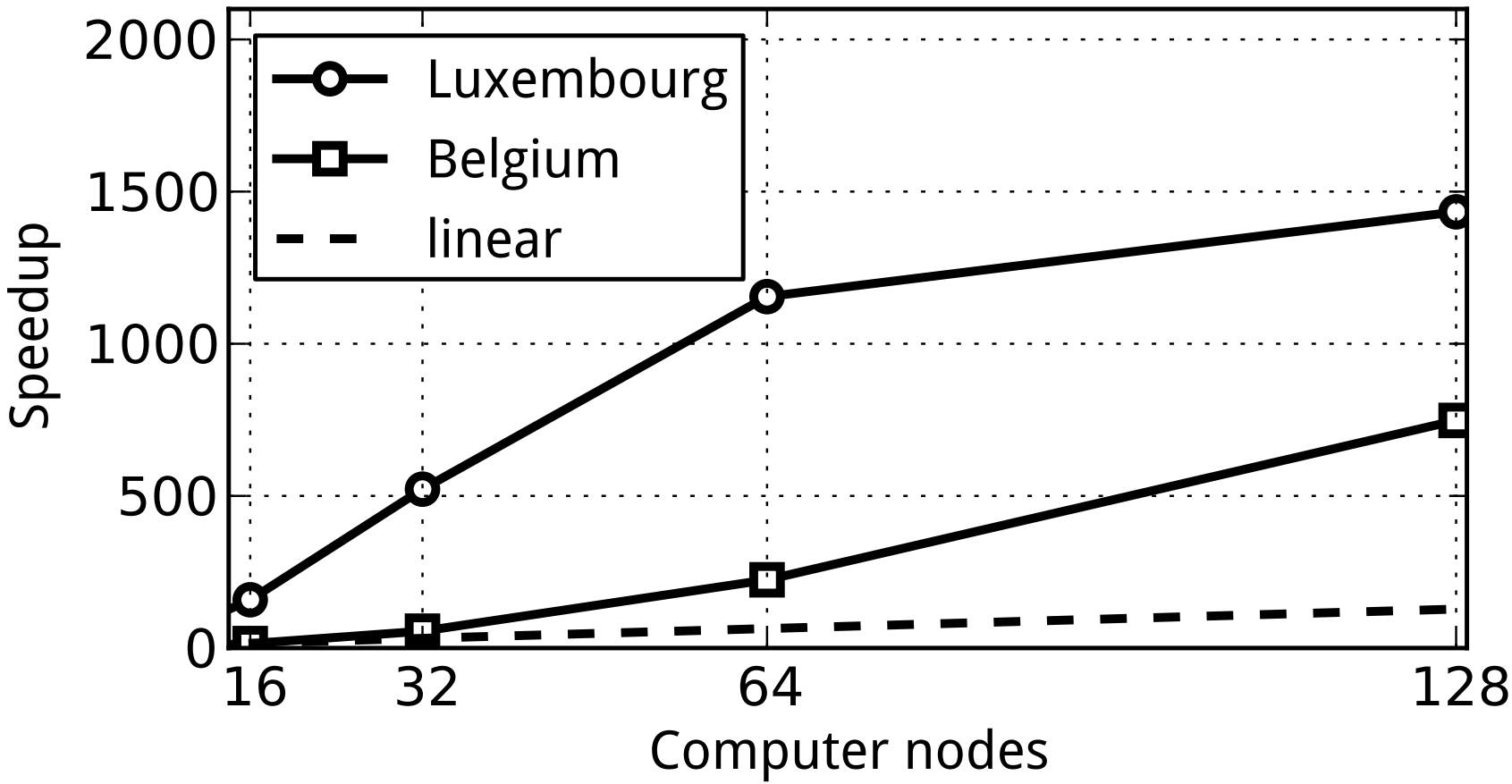
We performed some experiments to evaluate our approach using 128 nodes of a cluster in our university.
The graphs analysed were the road maps of Luxembourg and Belgium from OpenStreetMap. The contain about ... vertices and were distributed over the cluster nodes by this function.

The Belgium road map had to be coarsened 5 times to be able to use Chaco.

Execution time

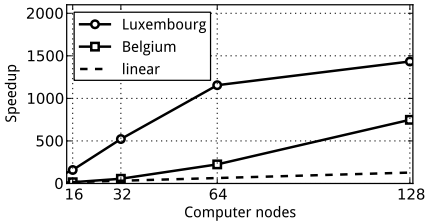


Speedup


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

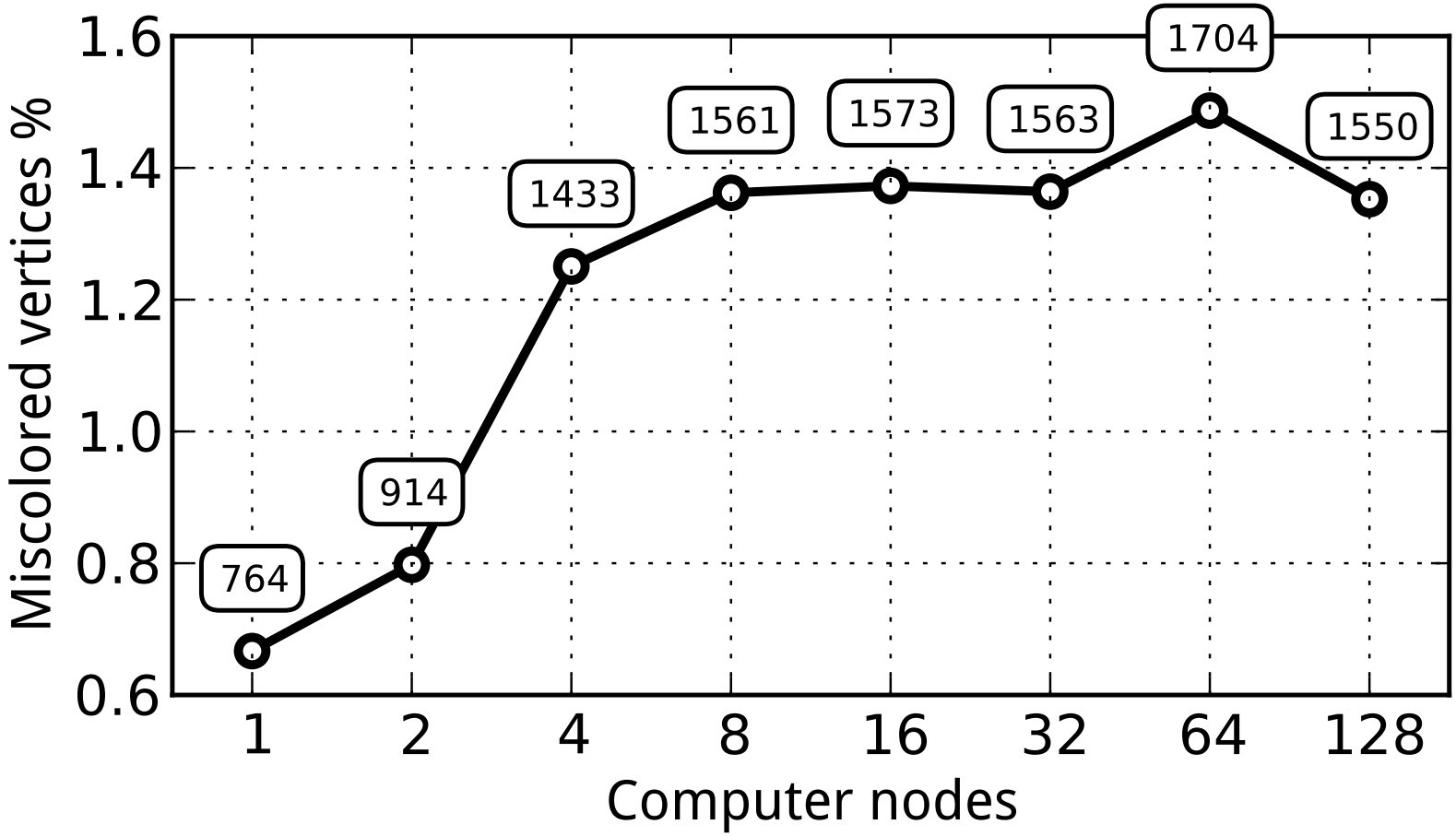
Very Large GraphPartitioningby Means ofParallel DBMS

Speedup



The speedup looks superlinear because we distribute the problem that has cubic complexity and ignore the relations between the distributed parts. This leads to some degree of error, that is demonstrated on the next slides.

Quality (Luxembourg)

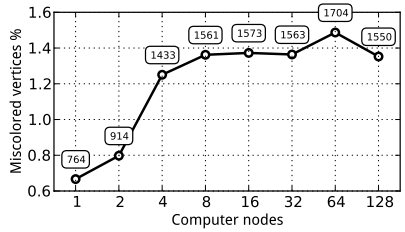


Random partitioning gives 30 % miscolored vertices.

Very Large GraphPartitioningby Means ofParallel DBMS

Quality (Luxembourg)

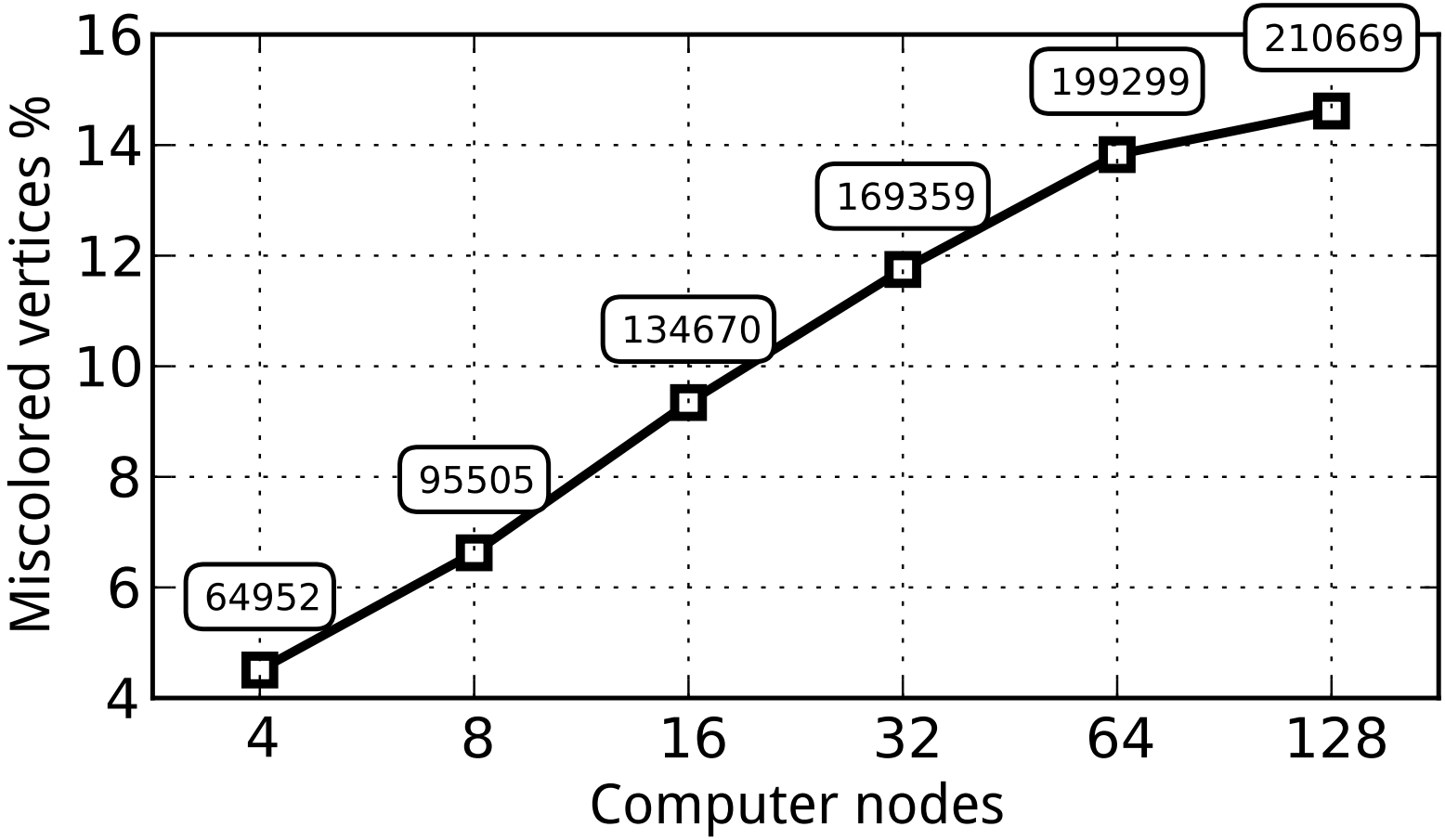
Quality (Luxembourg)



Random partitioning gives 30 % miscolored vertices.

The miscolored vertices are the vertices that would give a better result if we moved them to the opposite partition.

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