Generating Plan Diagrams For High Dimensions and Higher Resolution

A PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
Master of Engineering
IN
Computer Science & Engineering

BY

Sai Sandeep Balbari

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

June, 2017

Declaration of Originality

I, **Sai Sandeep Balbari**, with SR No. **04-04-00-10-41-15-1-12297** hereby declare that the material presented in the thesis titled

Generating Plan Diagrams For High Dimensions and Higher Resolution

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2015-2017**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: **Prof. Jayant R. Haritsa** Advisor Signature

DEDICATED TO

My Family, Teachers and Friends

for their love and support

Acknowledgements

I wish to express my gratitude to my advisor, Professor Jayant R. Haritsa for his generous advice, inspiring guidance and encouragement throughout my research for this work. This thesis could not have been completed without his professional guidance. I have been extremely lucky to work with him.

I am thankful to Srinivas Karthik for his assistance and guidance. It has been a great experience to work with him. I thank my fellow labmates in Database Systems Lab: Davinder Singh, Priyanka Sharma, Raghav Sood, Vinay Rijhwani, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last two years.

I would like to thank the Department of Computer Science & Automation for providing a wonderful learning experience and excellent study environment. Also I thank the IISc staff and friends for making my stay at IISc a great learning experience. I am also thankful to my best friends Rahul Raj Kumar and Vikas Naik for all the memorable moments at IISc.

Finally, I express my wholehearted gratitude to my parents for their financial aid and lasting support without question.

Abstract

Modern database systems use a query optimizer to identify the most efficient strategy called "plan", to execute declarative SQL queries. Optimization is a mandatory because the difference between the cost of the best plan and a random choice could be in orders of magnitude. For a query, on a given database and system configuration, the optimizer's plan choice is primarily a function of the selectivities of base predicates and join predicates of the relations present in the query. A pictorial enumeration of the execution plan choices of a database query optimizer over the relational selectivity space is called a "plan diagram". **"Picasso"** [6] tool is a database query optimizer visualizer that enables us to investigate plan diagrams.

The plan choices made by the optimizer are called parametric optimal set of plans (POSP). The bouquet identification phase in **"Plan Bouquet"** [3] and the plan diagram generation in **"Picasso"** [6] can be done only after the complete POSP is identified over the entire error-prone selectivity space (ESS). The time required to produce the POSP for the ESS increases exponentially with the dimensionality of the space by increasing the number of error-prone predicates for a given query. In "Robust Query Processing" [5], the POSP identification phase for queries having error-prone base predicates was improved by massive parallelization.

Once the POSP result is generated after parallelization, verification is done to check if it is correct or not using **"Picasso"**. In this work, we have generated the **"Picasso Packets"** from the raw results obtained after parallelization for queries having high dimensional error-prone base predicates (4) for higher resolution (100). Experiments were carried out with a suite of multi-dimensional TPC-H query templates on the PostgreSQL optimizer. The verification demonstrates that the results obtained by parallelization are correct and thus we have developed a parallelized version of **"Picasso"** to generate complex plan diagrams in significantly less time. For a query having 4 dimensions and 100 resolution, we were able to reduce time taken to generate plan diagrams from 166 hours to 3 hour 20 minutes. Further, changes were made to **"Parallelized Picasso"** to reduce the **"Picasso Packet"** size without affecting its behaviour. The parallelization phase of POSP generation is now extended from queries having error-prone base predicates to queries having error-prone join predicates.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In database systems, whenever a query is fired, generating query result involves finding a query plan during compile-time and executing the same query plan during run-time. Query plan is an ordered sequence of steps to fetch the results of an SQL query. For a particular query, all the query plans generated by the optimizer give the same query result. But, the cost of all query plans may not be the same. The query plan with least cost is chosen as the best query plan. When a query is executed, the best plan is passed to the executor module to generate the results which are shown as output.

The behaviour of modern query optimizers is seen from a fresh perspective by the introduction and development of **"Plan Diagram"** concept. A plan diagram is a visual representation of the plan choices made by the optimizer over a space of input parameters, such as relational selectivities. In short, plan diagrams pictorially capture the geometries of the optimality regions of the parametric optimal set of plans (POSP).

1.1 Motivation

QUEST [4] (QUery Execution without Selectivity estimation) is a Java based prototype implementation of **"Plan Bouquet"** [3] technique. It visually shows bouquet execution process and provides interactivity during execution. QUEST includes database system with incorporated features required for plan bouquet. Currently, PostgreSQL is the database system used with QUEST wherein all required features are implemented.

In "Robust Query Processing" [5], *join-selectivity injection* feature was implemented in a database to evaluate a query with error-prone *join* predicates. By doing this, we can get the optimizer's optimal plan for a given join selectivity. POSP identification was done for queries where *base* predicates are error-prone. This is now extended to queries having multi-dimensional error-prone *join* predicates and also error-prone *base* predicates.

PICASSO [6] is also a Java based implementation that gives a visual representation of the plan choices made by the optimizer over an input parameter space, whose dimensions may include database, query and system-related features.

In order to make both QUEST and PICASSO practically useful for queries having multi-dimensional error-prone predicates, we need to reduce the POSP generation time. For a query, the time required to produce the POSP for the entire ESS increases exponentially with the dimensionality of the ESS. If the resolution of a d-dimensional ESS is r, then the number of optimization calls made for producing the POSP is r^d, which is computationally expensive. In [5], a parallel algorithm was proposed to find the POSP for queries having multi-dimensional error-prone base predicates.

1.2 Contribution

In this work, we have created a **Picasso Packet** from the POSP generated after parallelization. We have also optimized, automated and extended the parallelized POSP generation phase on delta-cluster having 64 cores. We have also modified PICASSO so that the resulting "Picasso packets" occupy less storage space. Our contribution can be divided into three categories.

1.2.1 Parallelization Modifications

Here we explain the modifications done to the parallelization code for multi-core clusters which use PostgreSQL 9.4 database engine which has selectivity injection feature.

1.2.2 Parallelized Picasso

Here, we show how the raw results obtained from parallelization are used to create a picasso packet which can be loaded to show the plan diagrams. We will also show in the experiments how the plan diagrams generated from these picasso packets look like.

1.2.3 PICASSO Modifications

Here, we explain the modifications done to PICASSO so that the picasso packets that were created from raw results of parallelization can be saved in less storage space.

1.3 Organization

Section 2 provides an overview of POSP generation. Section 3 shows the existing PICASSO architecture. Parallelized PICASSO is explained in Section 4. In Section 5 we show the empirical analysis of our approach. Section 6 gives the conclusion and discusses future work.

Chapter 2

POSP Generation

In this chapter, we present necessary details about POSP generation and its applications.

2.1 Serial processing for POSP generation

For a given query, if we increase the error-prone dimensions, the time required to generate the POSP for the entire ESS will increase exponentially. If the resolution of a d-dimensional ESS is r, then the number of optimization calls made for producing the POSP is r^d, which is computationally expensive. Serial processing to generate this POSP takes a lot of time.

For a query with 4 error-prone base predicates with 100 resolution, the total number of optimization calls made for producing POSP will be 10^8.

2.2 Parallelization of POSP generation

In "Robust Query Processing" [5], a parallel algorithm was proposed that reduced the POSP generation time for error-prone base predicates. In this implementation, we need to supply the constant values of error-prone base predicates of the query.

Collection of constant values for a specified "error-prone base predicate" for a given "query", "selectivity distribution", "resolution" is done as follows: Collect the constants corresponding to all the error-prone base predicates of the query by repeating the above process for the respective error-prone base predicates.

These constants are supplied during the parallelization phase of POSP generation

2.2.1 Parallelization Limitations and Modifications

In "Robust Query Processing" [5], the constant values for error-prone base predicates are supplied manually. These values vary depending on the query, error-prone predicates, selectivity distribution, resolution. Hence it is very difficult to collect the constants by running the **"Pi-**

casso" tool manually each time when there is a change in either "selectivity distribution" or "resolution" or "query" or "error-prone predicate".

This problem is solved by using PostgreSQL 9.4 database engine with selectivity injection feature where we can inject the selectivities corresponding to error-prone predicates in the query while making optimization calls to the database. These selectivities for a given "selectivity distribution", "resolution" are calculated once and can be used for all the queries for all the error-prone predicates (*base* as well as *join*).

```
SELECT   *
FROM     lineitem, orders, part
WHERE    p_partkey = l_partkey
         and l_orderkey = o_orderkey
         and p_retailprice < 1000;
```

Example SQL Query (EQ)

This implementation of POSP generation in "Robust Query Processing" [5] is also extended from error-prone base predicates (like *p_retailprice* $<$ *1000* in EQ) to error-prone join predicates(like *l_orderkey* $=$ *o_orderkey* in EQ). Changes were made to this implementation to use selectivity injection feature that was added in PostgreSQL 9.4.

```
SELECTIVITY (p_retailprice < 1000
             and l_orderkey = o_orderkey)
             (0.3, 0.7)
SELECT   *
FROM     lineitem, orders, part
WHERE    p_partkey = l_partkey
         and l_orderkey = o_orderkey
         and p_retailprice < 1000;
```

Example Query with Selectivity Injection (EQ2)

In EQ2, *p_retailprice* $<$ *1000* and *l_orderkey* $=$ *o_orderkey* are the error-prone predicates with 0.3, 0.7 as their corresponding injected selectivities. Therefore, the entire process of generating the POSP results is automated in the new implementation.

The advantages of using Selectivity Injection are described in the form of a table by comparing it with the existing approach to generate POSP

Parallelization Code for Base Predicates	**Parallelization Code with Selectivity Injection**
works only if base predicates are error-prone	works for both, base and join error-prone predicates
Constants values need to be supplied with the Query	Selectivity values need to be supplied with the Query
Picasso tool has to run to get the constants	No need to run Picasso tool to get the selectivities
Constants change by changing "Selectivity distribution", "Resolution"	Selectivities change by changing "Selectivity distribution", "Resolution"
Constants change by changing "Query", "Error-Prone Predicates"	Selectivities do not change by changing "Query", "Error-Prone Predicates"
This code is partially automated	This code is completely automated

Table 2.1: Previous implementation v/s Current implementation for POSP generation

The output results were storing Plan Number, Cost, Cardinality at each selectivity location in text format needing large storage space to save them. For a query having 4 dimensions and 100 resolution, the output results needed 2.5GB of storage space. As we increase the dimension to 5 by adding one more error-prone base predicate, the storage space needed to save the results will go up to 250GB. It becomes very difficult to handle such large files.

So, the output results are now stored in the binary format instead of text format there by reducing the storage space needed to save them. By doing this, we are able to storage the POSP results of parallelization in 40% of the space that was used before.

2.3 Applications

The POSP generated for high dimensions and higher resolution can be used for pre-processing phase of **"Plan Bouquet"** approach for robust query processing and to visualize complex plan diagrams in **"Picasso"** Database Query Optimizer Visualizer [6].

Chapter 3

Picasso Database Query Optimizer Visualizer

"Picasso" [6], is a tool that gives a pictorial enumeration of the execution plan choices of a database query optimizer over the relational selectivity space. It is completely written in Java and, in principle, operates in a platform independent manner.

3.1 Existing Architecture

There are three processes involved in a Picasso setup in addition to a database as shown in Figure 1.

1. **Picasso Client:** Users enter query templates and visualize the associated diagrams through this.
2. **Picasso Server:** Converts the query templates into the equivalent set of query instances to submit to the database engines and gathers the associated execution plans
3. **Database Engine:** Connects to the database and produces the execution plans for the queries.
4. **Database:** The entire data is stored here. Also, the new tables created by picasso are also stored here.

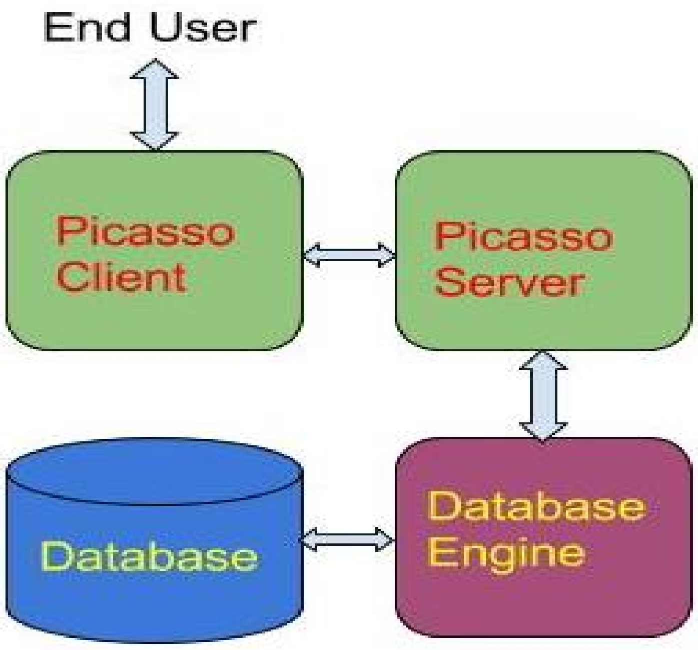


Figure 3.1: **Picasso Architecture**

The three processes, Picasso Client, Picasso Server, Database Engine can all execute on the same machine or on different machines. Further, multiple Picasso clients can connect to a single Picasso server, which in turn can connect to multiple database engines.

Chapter 4

Parallelized PICASSO

In this section, we elaborate the procedure to generate "Picasso Packets" using parallelization which can be loaded to visualize the plan diagrams.

4.1 Collection of Raw Results

For a given multi-dimensional query template, selectivity space resolution, distribution, using parallelization [5] code written in MPI and C++, we generate POSP in the form of raw results. These results contain Plan Number, Cost, Cardinality stored for each combination of selectivities of error prone base predicates. The plan trees and query template are separately stored in text format. The information regarding the number of error-prone predicates, resolution of selectivity space, maximum and minimum values of cost and cardinality and total number of plans are collected separately.

4.2 Creation of Picasso Packet

From these raw results, a packet is created which will be similar to the picasso packet that would be generated without parallelization if the same multi-dimensional query template with same selectivity space resolution and distribution were given to the existing PICASSO tool.

4.2.1 Implementation Details

In this section, we explain the implementation details of creating picasso packets

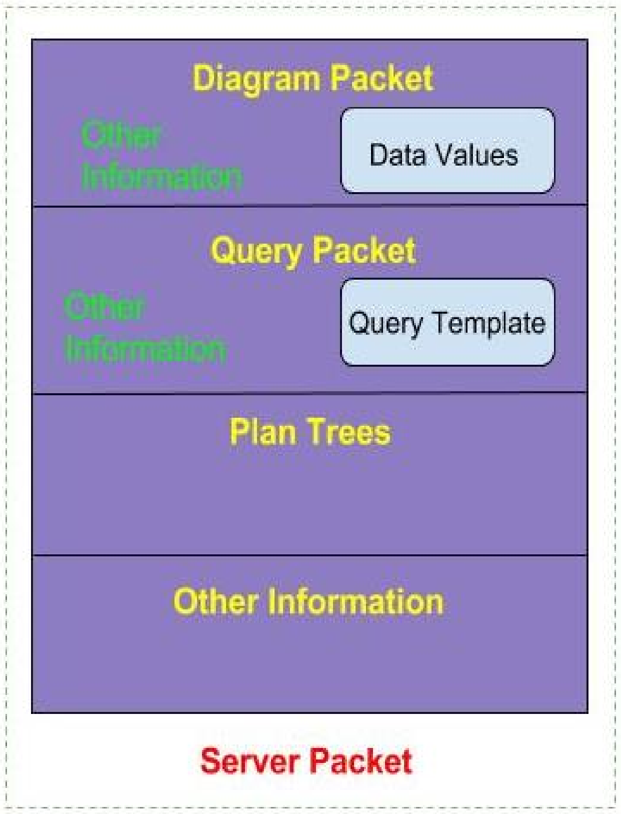


Figure 4.1: **Picasso Packet Structure**

“Picasso Packet” is a “Server Packet” which is given by the “Picasso Server”. It contains “Diagram Packet”, “Query Packet” “Plan Trees” and some additional information regarding clientId, Port Number, Connection information.

The structure of Picasso Packet looks as shown in Figure 2 where we explain three important fields:

1. **Data Values** in “Diagram Packet” which stores Plan Number, Cost, Cardinality and isRepresentative for each selectivity combination.

2. **Query Template** in “Query Packet” which contains the multi-dimensional query template

3. **Plan Trees** in “Server Packet” where all the plan trees are stored

Using the dimension and resolution information from the raw results, we first create an empty picasso packet using a new module called “Picasso Packet Creator”. This packet will have the same structure as that of the original picasso packet that would have been created when picasso tool was run on single-core system for the same query template with same dimension and resolution.

Now, we copy the plan number, cost, cardinality and isRepresentative at each selectivity location into the corresponding position in the empty packet. The field "isRepresentative" is set to **"false"** for all the points initially. The plan trees, query template, total number of plans, maximum and minimum values of cost and cardinality are also copied. Finally, it is stored on to the disk in the form of picasso packet which can be loaded using "load packet" feature present in PICASSO to visualize the plan diagrams.

4.3 Picasso Modifications

The "Data Values" structure present in "Diagram Packet" structure of "Picasso Packet" has two more additional fields "FPCdone", "succProb" which are used to do interpolation when Foreign Plan Cost (FPC) is used. In our experiments, we have not used FPC because of which, the values of these fields are same for all the points in the selectivity space. So, instead of storing the same information for all the points, we are storing them only once in the picasso packet. By doing this, we are able to save 5 Bytes of storage space for every selectivity combination of error prone predicates. Hence, we are able to generate the picasso packets occupying lesser storage space without losing any information.

4.4 Architecture of Parallelized Picasso

The latest architecture of Parallelized Picasso is shown in Figure 3 which operates in three phases:

1. generation of POSP in raw form using parallelization on multi-core clusters (Implemented in MPI and C++).

2. Creating a "Picasso Packet" from the POSP generated in raw form (Implemented in JAVA).

3. This "Picasso Packet" can be loaded to visualize the plan diagrams.

The GUI is modified and a query with 5 dimensional error-prone base predicates is run for 10 resolution for the first time.

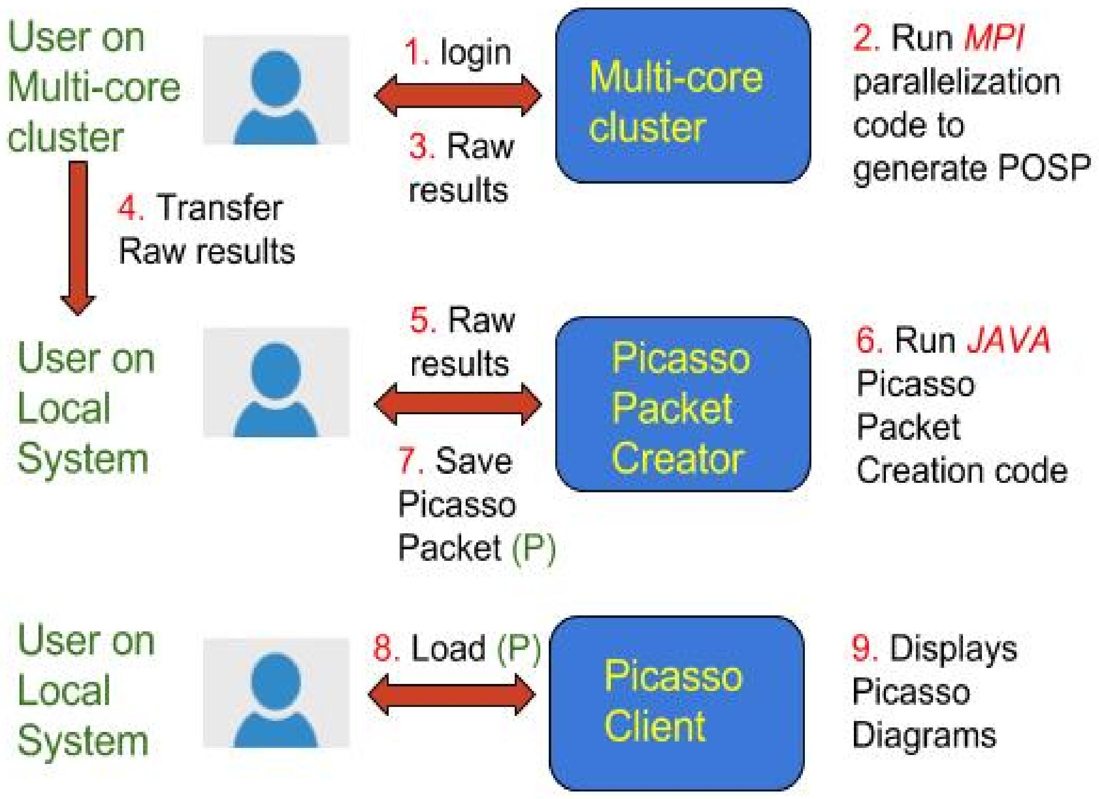

User on Multi-core cluster
1. login
3. Raw results
Multi-core cluster
2. Run MPI parallelization code to generate POSP
4. Transfer Raw results
User on Local System
5. Raw results
7. Save Picasso Packet (P)
Picasso Packet Creator
6. Run JAVA Picasso Packet Creation code
User on Local System
8. Load (P)
Picasso Client
9. Displays Picasso Diagrams

Figure 4.2: **Parallelized Picasso Architecture**

Chapter 5

Empirical Analysis

We used PostgreSQL 9.4 which has selectivity injection feature added in it. Our experiments were carried out on Delta Cluster composed of 4 nodes [1]. Each having 16 cores, 128GB RAM and they are connected using Infiniband Card for MPI communication and Dual-port Gigabit Ethernet Connectivity for enabling log-ins. The operating system used is CentOS 6.2 which is built on Linux x86_64 Platform. For our parallel algorithm implementation, we used GNU compiler collection with MVAPICH2 Message Passing Interface and for *Picasso Packet Creation*, we have used Java. We have performed all the experiments on TPC-H 5GB database [2].

5.1 Impact of Parallel Algorithm

We have used 4 nodes and 12 cores in each node for our experimental purpose. We got a speed-up of 48 times to generate the POSP in the form of raw results.

For the query having 4 dimensions and 100 resolution, we were able to generate the raw results in 2 hour 50 minutes. Next, the picasso packets were created in 30 minutes. Therefore, we were able to create the “Picasso Packet” in 3 hour 20 minutes. The existing PICASSO tool which runs on single-core systems takes 166 hours to generate the same picasso packets.

5.2 Verification through Picasso Diagrams

Here, we see that, when picasso packets generated from parallelization and the picasso packets generated by existing PICASSO for same query and same distribution and same resolution are loaded, the PICASSO diagrams are found to be

1. **Exactly Same** for low resolution from parallelization and for packets of low resolution generated from existing PICASSO.

2. **Similar** for packets of high resolution generated from parallelization and for packets of low resolution generated on existing PICASSO.

In all the experiments, the PostgreSQL configuration file retained the same.

5.3 Results

Here we see the impact of parallelization over queries having error-prone join predicates and the verification of POSP results from the picasso diagrams generated for the error-prone base predicates.

5.3.1 Parallelization for Join Predicates

For Queries where *join-predicates* are error-prone, experiments are performed on Query 5 of TPC-H 5GB database.

In all the queries shown below, the error prone join predicates are shown in bold and their corresponding selectivity values will vary from 0 to 1.

selectivity ***(c_custkey = o_custkey, o_orderkey = l_orderkey) (varies, varies)*** *select n_name, l_extendedprice, l_discount as revenue from customer, orders, lineitem, supplier, nation, region where c_custkey = o_custkey and l_orderkey = o_orderkey and l_suppkey = s_suppkey and s_nationkey = n_nationkey and n_regionkey = r_regionkey and o_totalprice <= 5000 and s_acctbal <= 10000;*

Query 5 with 2 error-prone join predicates

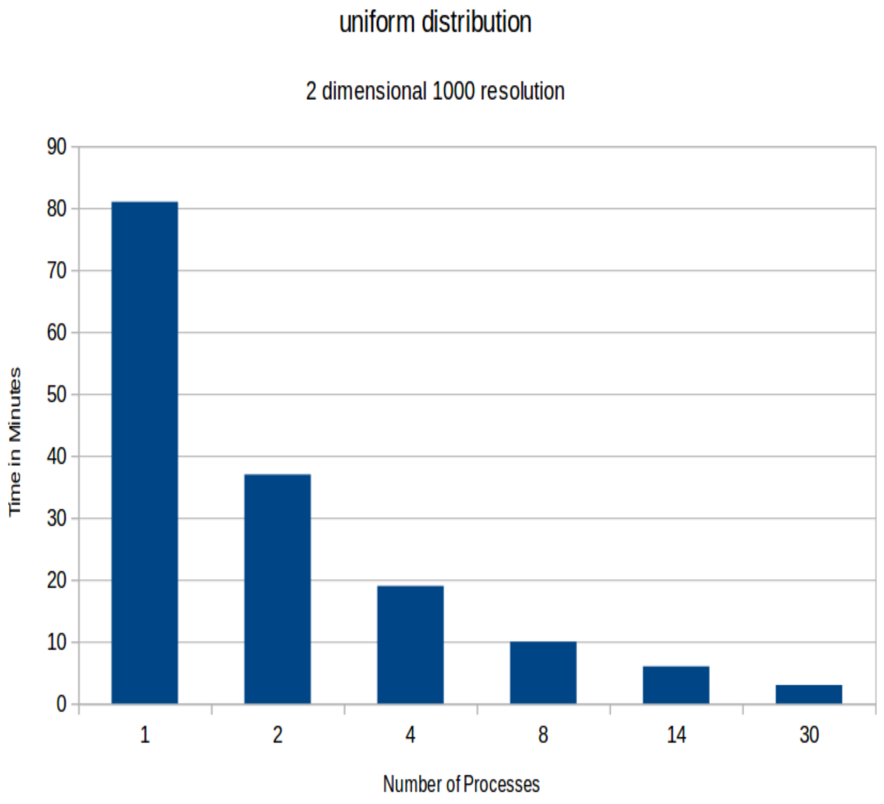


Figure 5.1: POSP Performance I

In Figure 4, we reduced the identification of POSP from 82 minutes to 3 minutes using 30 cores. Number of Plans in POSP = 6.

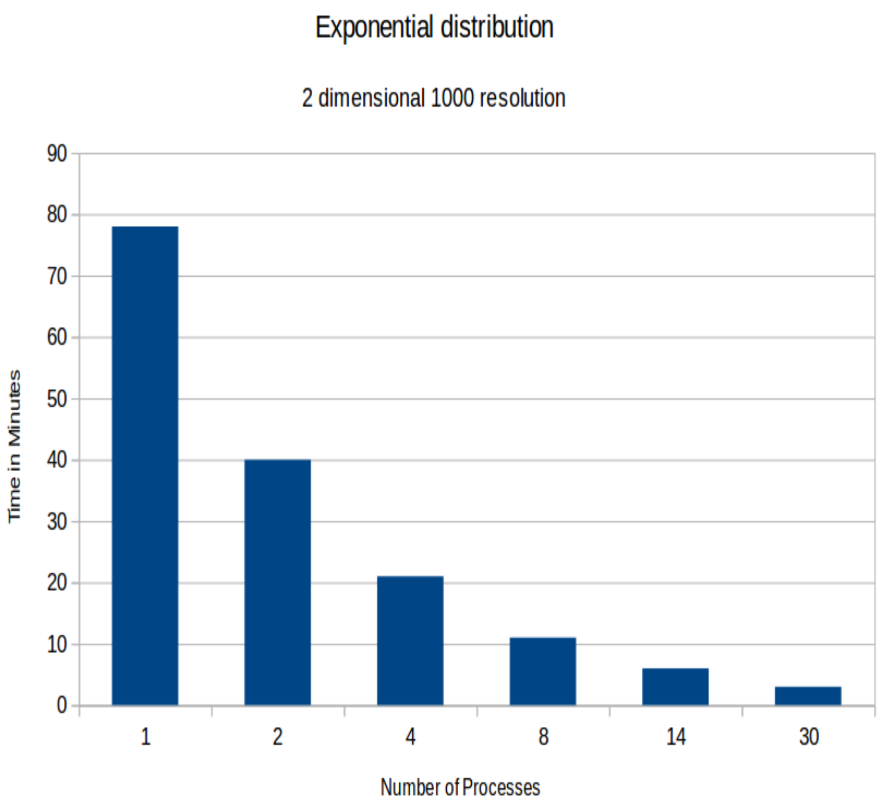


Figure 5.2: POSP Performance II

In Figure 5, we reduced the identification of POSP from 78 minutes to 3 minutes using 30 cores. Number of Plans in POSP = 8.

selectivity ***(c_custkey = o_custkey, o_orderkey = l_orderkey, s_suppkey = l_suppkey) (varies, varies, varies)*** *select n_name, l_extendedprice, l_discount as revenue from customer, orders, lineitem, supplier, nation, region where c_custkey = o_custkey and l_orderkey = o_orderkey and l_suppkey = s_suppkey and s_nationkey = n_nationkey and n_regionkey = r_regionkey and o_totalprice <= 5000 and s_acctbal <= 10000;*

Query 5 with 3 error-prone join predicates

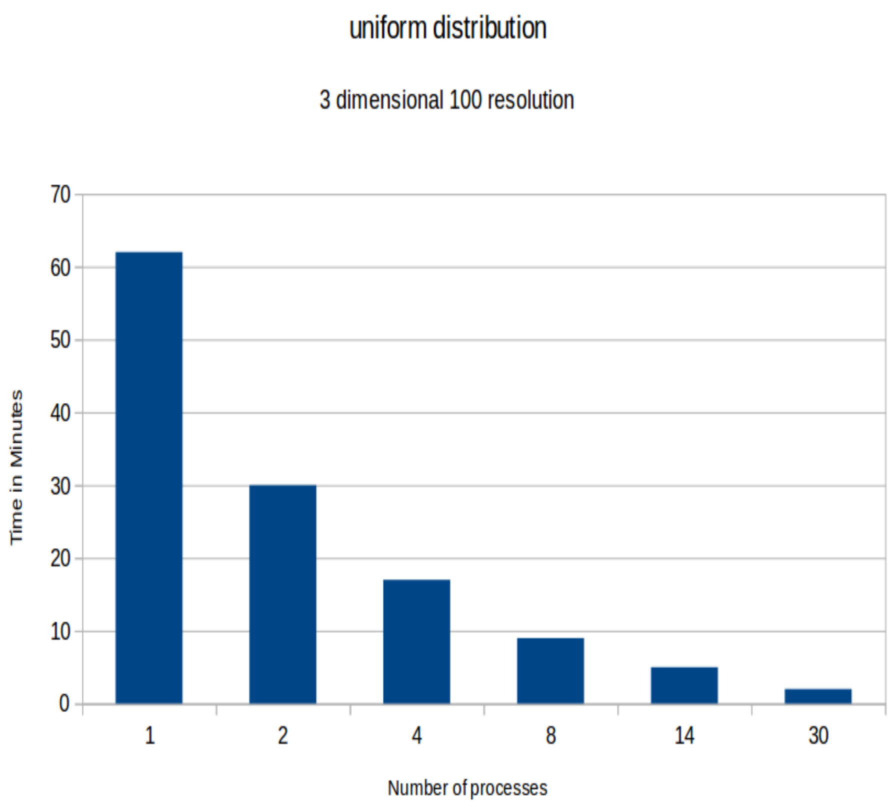


Figure 5.3: POSP Performance III

In Figure 6, we reduced the identification of POSP from 62 minutes to 2 minutes using 30 cores. Number of Plans in POSP = 13

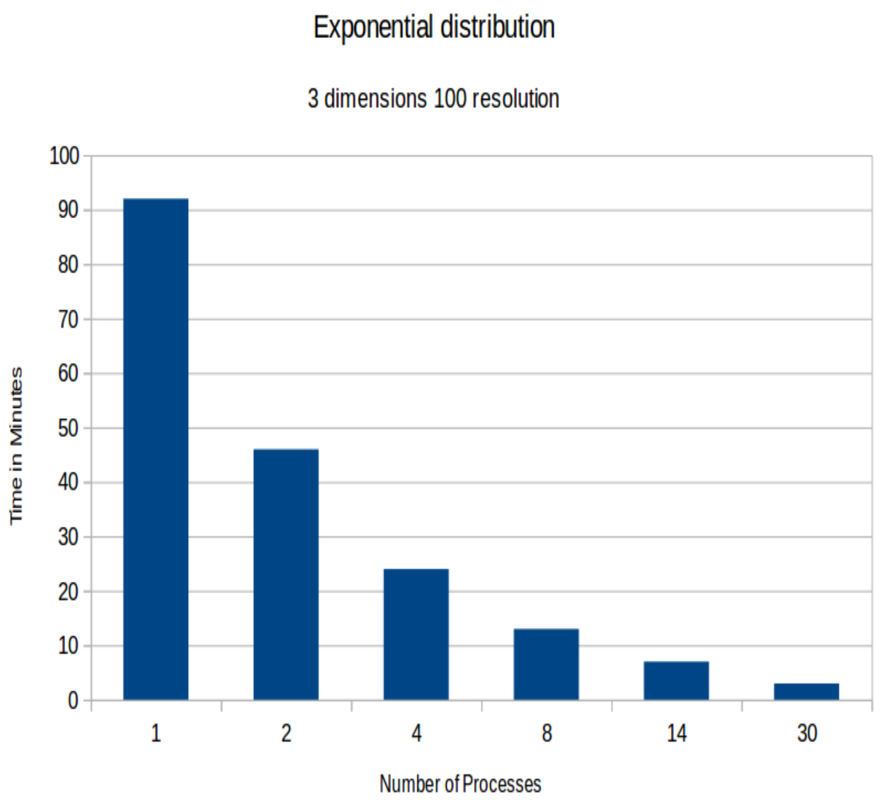


Figure 5.4: POSP Performance IV

In Figure 7, we reduced the identification of POSP from 92 minutes to 3 minutes using 30 cores. Number of Plans in POSP = 15.

For a query with 3 dimensions and 1000 resolution, the POSP can be generated in *16 hours* for exponential distribution and in *17 hours* for uniform distribution using 48 cores, which is very less when compared to POSP identification with the conventional single core approach which takes *30 days* approximately. It also reduces the compile-time overhead for plan bouquet to a great extent.

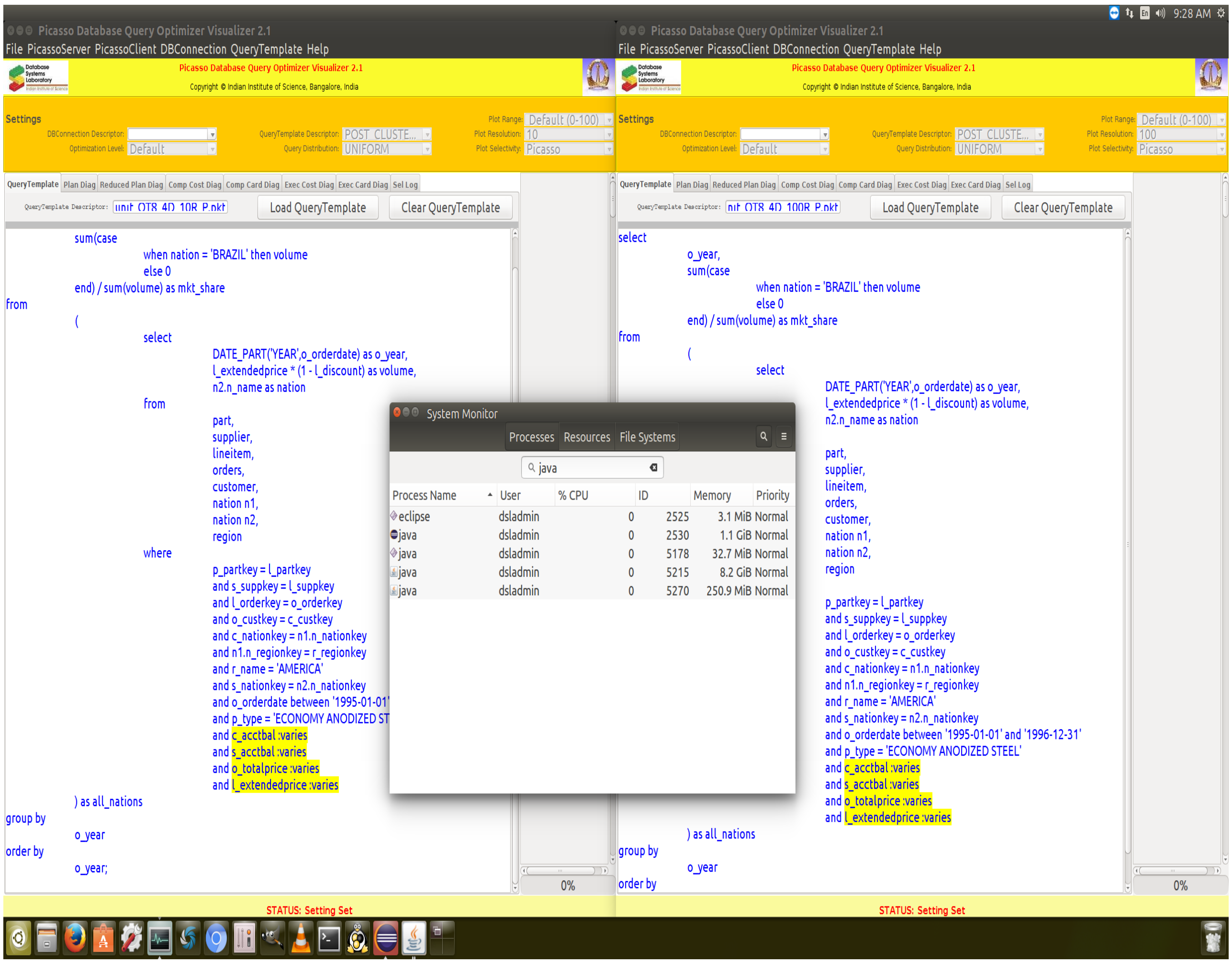
These POSP plans along with their costs and selectivity values can be used to run **"Plan Bouquet"** [3] and **"Spill Bound"** [7] on high dimensions and higher resolution on **"Quest"** [4]. **"Quest"** is a prototype system that showcases the concept of plan bouquet in existence.

5.3.2 Verification by Picasso Diagrams

In all the pictures shown, one half of the figure shows the PICASSO frame using PICASSO generated packet and the other half shows the PICASSO frame using packet generated from parallelized results.

We see that both the frames are showing same plan diagrams for low resolution.

The following figure shows that we are using same query template in both the frames


Picasso Database Query Optimizer Visualizer 2.1
File PicassoServer PicassoClient DBConnection QueryTemplate Help
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate Plan Diag Reduced Plan Diag Comp Cost Diag Comp Card Diag Exec Cost Diag Exec Card Diag Sel Log
Load QueryTemplate
Clear QueryTemplate
9:28 AM

Settings
DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

QueryTemplate Descriptor: unit_QT8_4D_10R_P.pkt

```
        sum(case
                when nation = 'BRAZIL' then volume
                else 0
        end) / sum(volume) as mkt_share
from
        (
                select
                        DATE_PART('YEAR',o_orderdate) as o_year,
                        l_extendedprice * (1 - l_discount) as volume,
                        n2.n_name as nation
                from
                        part,
                        supplier,
                        lineitem,
                        orders,
                        customer,
                        nation n1,
                        nation n2,
                        region
                where
                        p_partkey = l_partkey
                        and s_suppkey = l_suppkey
                        and l_orderkey = o_orderkey
                        and o_custkey = c_custkey
                        and c_nationkey = n1.n_nationkey
                        and n1.n_regionkey = r_regionkey
                        and r_name = 'AMERICA'
                        and s_nationkey = n2.n_nationkey
                        and o_orderdate between '1995-01-01'
                        and p_type = 'ECONOMY ANODIZED ST
                        and c_acctbal :varies
                        and s_acctbal :varies
                        and o_totalprice :varies
                        and l_extendedprice :varies
        ) as all_nations
group by
        o_year
order by
        o_year;
```

0%

STATUS: Setting Set

Settings
DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 100
Plot Selectivity: Picasso

QueryTemplate Descriptor: nit_QT8_4D_100R_P.pkt

```
select
        o_year,
        sum(case
                when nation = 'BRAZIL' then volume
                else 0
        end) / sum(volume) as mkt_share
from
        (
                select
                        DATE_PART('YEAR',o_orderdate) as o_year,
                        l_extendedprice * (1 - l_discount) as volume,
                        n2.n_name as nation

                        part,
                        supplier,
                        lineitem,
                        orders,
                        customer,
                        nation n1,
                        nation n2,
                        region

                        p_partkey = l_partkey
                        and s_suppkey = l_suppkey
                        and l_orderkey = o_orderkey
                        and o_custkey = c_custkey
                        and c_nationkey = n1.n_nationkey
                        and n1.n_regionkey = r_regionkey
                        and r_name = 'AMERICA'
                        and s_nationkey = n2.n_nationkey
                        and o_orderdate between '1995-01-01' and '1996-12-31'
                        and p_type = 'ECONOMY ANODIZED STEEL'
                        and c_acctbal :varies
                        and s_acctbal :varies
                        and o_totalprice :varies
                        and l_extendedprice :varies
        ) as all_nations
group by
        o_year
order by
```

0%

STATUS: Setting Set

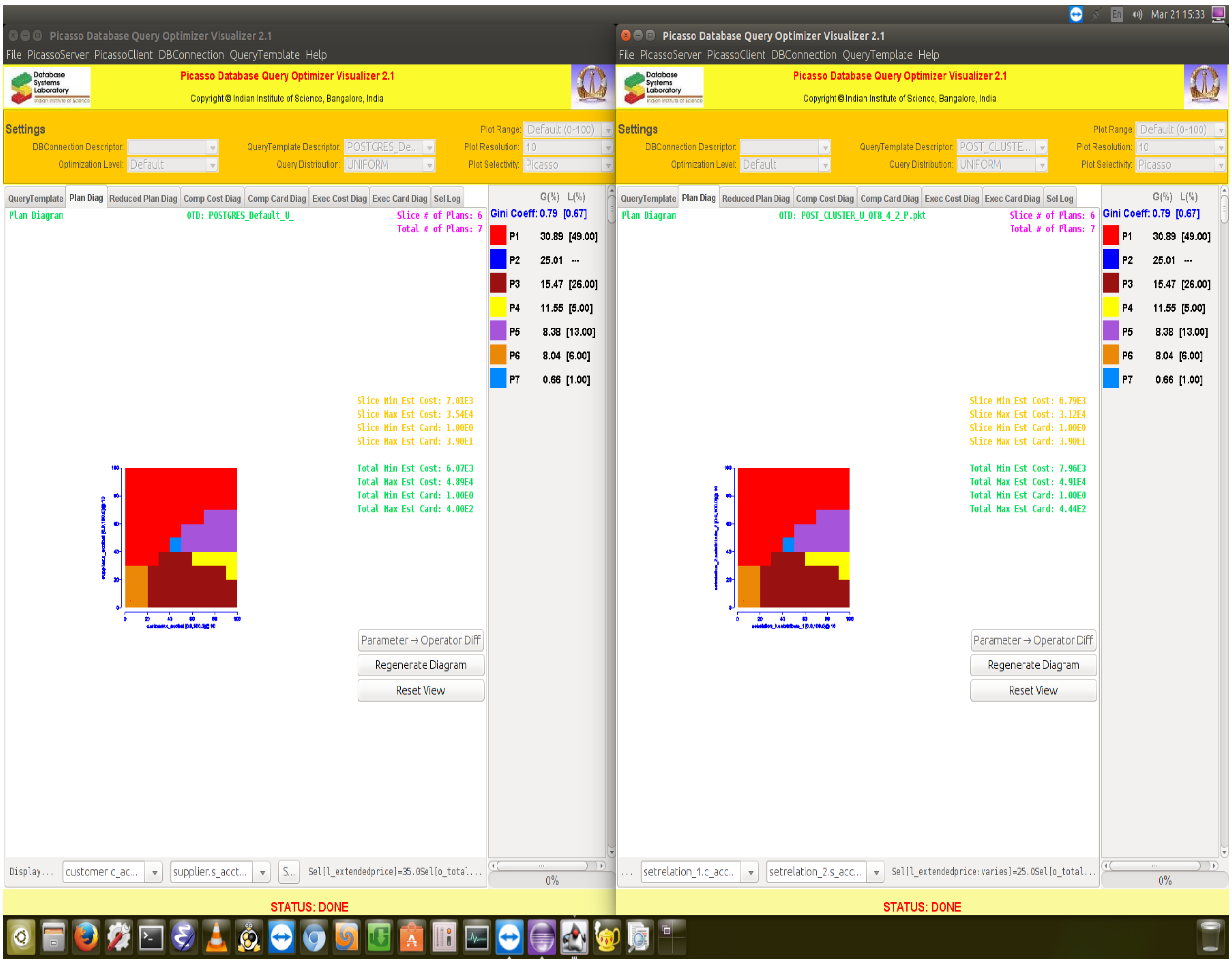
System Monitor
Processes Resources File Systems

java

```
Process Name      User       % CPU    ID      Memory   Priority
eclipse           dsladmin       0  2525     3.1 MiB   Normal
java              dsladmin       0  2530     1.1 GiB   Normal
java              dsladmin       0  5178    32.7 MiB   Normal
java              dsladmin       0  5215     8.2 GiB   Normal
java              dsladmin       0  5270   250.9 MiB   Normal
```

Figure 5.5: **TPC-H Query Q8**

The following figure shows that the plan diagrams are exactly same for the slices taken when the selectivities of *o_totalprice* is 25% and *l_extendedprice* is 35%.


Picasso Database Query Optimizer Visualizer 2.1

Settings
DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POSTGRES_De...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

Plan Diagram   QTD: POSTGRES_Default_U_
Slice # of Plans: 6
Total # of Plans: 7

		G(%)	L(%)
	Gini Coeff:	0.79	[0.67]
🟥	P1	30.89	[49.00]
🟦	P2	25.01	---
🟥	P3	15.47	[26.00]
🟨	P4	11.55	[5.00]
🟪	P5	8.38	[13.00]
🟧	P6	8.04	[6.00]
🟦	P7	0.66	[1.00]

```
Slice Min Est Cost: 7.01E3
Slice Max Est Cost: 3.54E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 3.90E1

Total Min Est Cost: 6.07E3
Total Max Est Cost: 4.89E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.00E2
```

Display... customer.c_ac... supplier.s_acct... S... Sel[l_extendedprice]=35.0Sel[o_total...

STATUS: DONE

Picasso Database Query Optimizer Visualizer 2.1

Settings
DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

Plan Diagram   QTD: POST_CLUSTER_U_QT8_4_2_P.pkt
Slice # of Plans: 6
Total # of Plans: 7

		G(%)	L(%)
	Gini Coeff:	0.79	[0.67]
🟥	P1	30.89	[49.00]
🟦	P2	25.01	---
🟥	P3	15.47	[26.00]
🟨	P4	11.55	[5.00]
🟪	P5	8.38	[13.00]
🟧	P6	8.04	[6.00]
🟦	P7	0.66	[1.00]

```
Slice Min Est Cost: 6.79E3
Slice Max Est Cost: 3.12E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 3.90E1

Total Min Est Cost: 7.96E3
Total Max Est Cost: 4.91E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.44E2
```

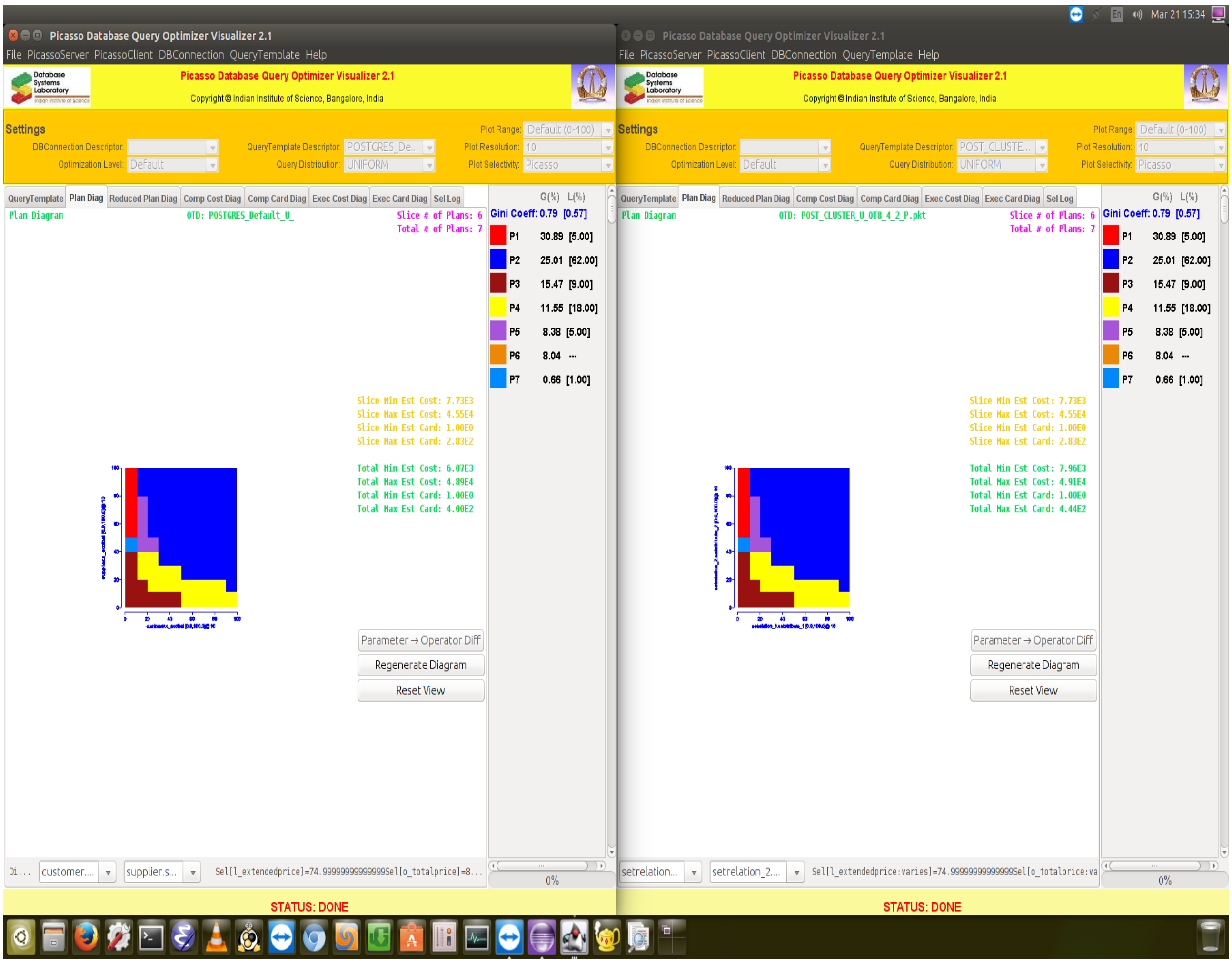
... setrelation_1.c_acc... setrelation_2.s_acc... Sel[l_extendedprice:varies]=25.0Sel[o_total...

STATUS: DONE

File PicassoServer PicassoClient DBConnection QueryTemplate Help
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate Plan Diag Reduced Plan Diag Comp Cost Diag Comp Card Diag Exec Cost Diag Exec Card Diag Sel Log
Parameter → Operator Diff
Regenerate Diagram
Reset View
0%
Mar 21 15:33

Figure 5.6: **Plan Diagram for o_25&l_35**

The following figure shows that the plan diagrams are exactly same for the slices taken when the selectivities of *o_totalprice* is 75% and *l_extendedprice* is 85%.


Picasso Database Query Optimizer Visualizer 2.1
File PicassoServer PicassoClient DBConnection QueryTemplate Help
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate Plan Diag Reduced Plan Diag Comp Cost Diag Comp Card Diag Exec Cost Diag Exec Card Diag Sel Log
Parameter → Operator Diff
Regenerate Diagram
Reset View
Mar 21 15:34

Settings

DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POSTGRES_De...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

Plan Diagram   QTD: POSTGRES_Default_U_   Slice # of Plans: 6   Total # of Plans: 7

		G(%)	L(%)
Gini Coeff:		0.79	[0.57]
🟥	P1	30.89	[5.00]
🟦	P2	25.01	[62.00]
🟫	P3	15.47	[9.00]
🟨	P4	11.55	[18.00]
🟪	P5	8.38	[5.00]
🟧	P6	8.04	---
🟦	P7	0.66	[1.00]

```
Slice Min Est Cost: 7.73E3
Slice Max Est Cost: 4.55E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 2.83E2

Total Min Est Cost: 6.07E3
Total Max Est Cost: 4.89E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.00E2
```

Di... customer.... supplier.s... Sel[l_extendedprice]=74.99999999999999Sel[o_totalprice]=8...   0%

STATUS: DONE

Settings

DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

Plan Diagram   QTD: POST_CLUSTER_U_QT8_4_2_P.pkt   Slice # of Plans: 6   Total # of Plans: 7

		G(%)	L(%)
Gini Coeff:		0.79	[0.57]
🟥	P1	30.89	[5.00]
🟦	P2	25.01	[62.00]
🟫	P3	15.47	[9.00]
🟨	P4	11.55	[18.00]
🟪	P5	8.38	[5.00]
🟧	P6	8.04	---
🟦	P7	0.66	[1.00]

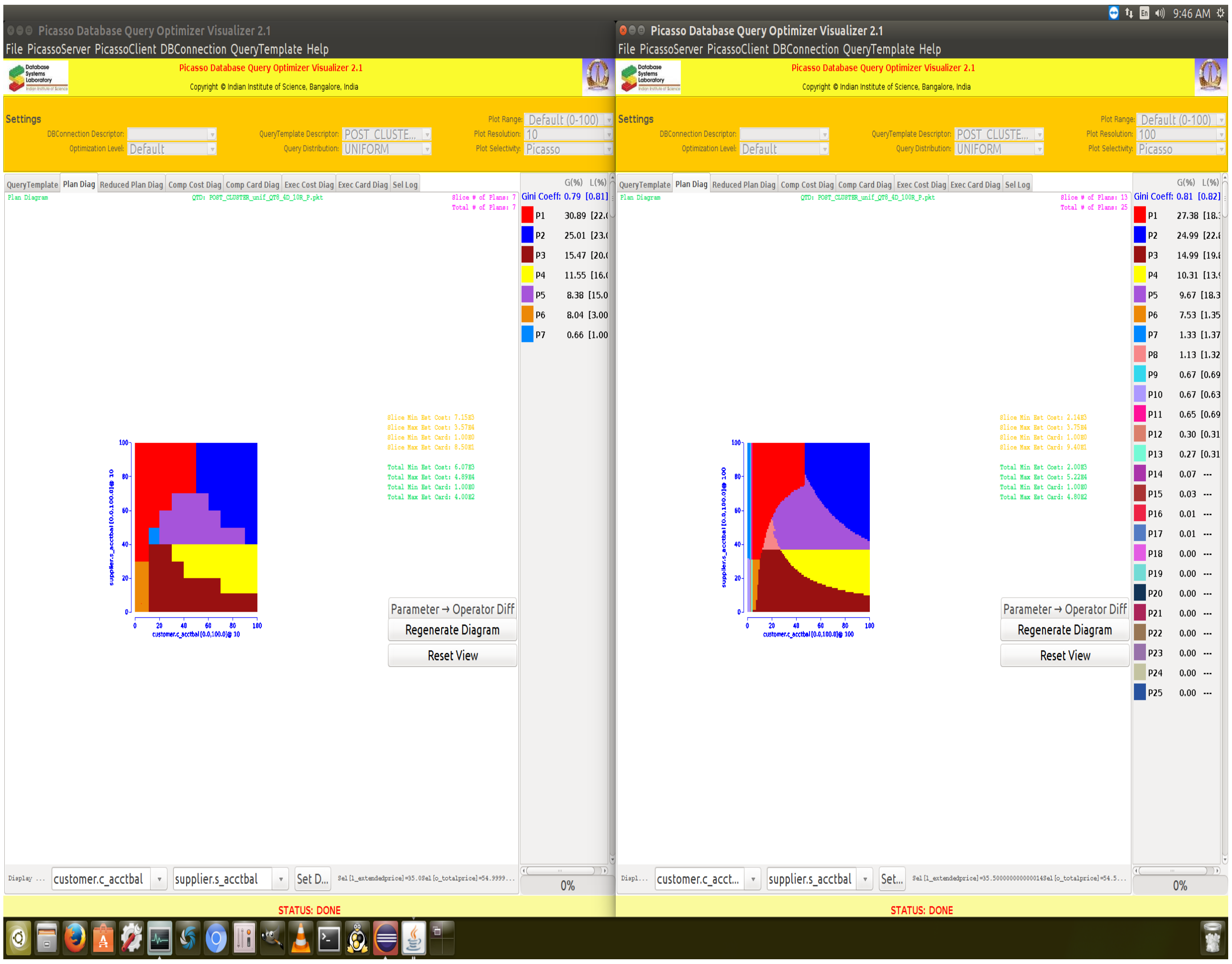
```
Slice Min Est Cost: 7.73E3
Slice Max Est Cost: 4.55E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 2.83E2

Total Min Est Cost: 7.96E3
Total Max Est Cost: 4.91E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.44E2
```

setrelation... setrelation_2.... Sel[l_extendedprice:varies]=74.99999999999999Sel[o_totalprice:va   0%

STATUS: DONE

Figure 5.7: **Plan Diagram for o_75&l_85**

The following 3 figures show that the plan diagram, cardinality diagram and cost diagram are similar for the slices taken when the selectivities of *o_totalprice* is 35.5% and *l_extendedprice* is 54.5%



Picasso Database Query Optimizer Visualizer 2.1

Copyright © Indian Institute of Science, Bangalore, India

Settings

	Left window	Right window
DBConnection Descriptor:		
Optimization Level:	Default	Default
QueryTemplate Descriptor:	POST_CLUSTE...	POST_CLUSTE...
Query Distribution:	UNIFORM	UNIFORM
Plot Range:	Default (0-100)	Default (0-100)
Plot Resolution:	10	100
Plot Selectivity:	Picasso	Picasso

Left window:

QTD: POST_CLUSTER_unif_QT8_4D_10R_P.pkt

Slice # of Plans: 7
Total # of Plans: 7

```
Slice Min Est Cost: 7.15E3
Slice Max Est Cost: 3.57E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 8.50E1

Total Min Est Cost: 6.07E3
Total Max Est Cost: 4.89E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.00E2
```

		G(%)	L(%)
Gini Coeff:		0.79	[0.81]
🟥	P1	30.89	[22.0
🟦	P2	25.01	[23.0
🟫	P3	15.47	[20.0
🟨	P4	11.55	[16.0
🟪	P5	8.38	[15.0
🟧	P6	8.04	[3.00
🟦	P7	0.66	[1.00

Display ... customer.c_acctbal supplier.s_acctbal Set D... Sel [l_extendedprice]=35.0Sel [o_totalprice]=54.9999...

STATUS: DONE

Right window:

QTD: POST_CLUSTER_unif_QT8_4D_100R_P.pkt

Slice # of Plans: 13
Total # of Plans: 25

```
Slice Min Est Cost: 2.14E3
Slice Max Est Cost: 3.75E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 9.40E1

Total Min Est Cost: 2.00E3
Total Max Est Cost: 5.22E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.80E2
```

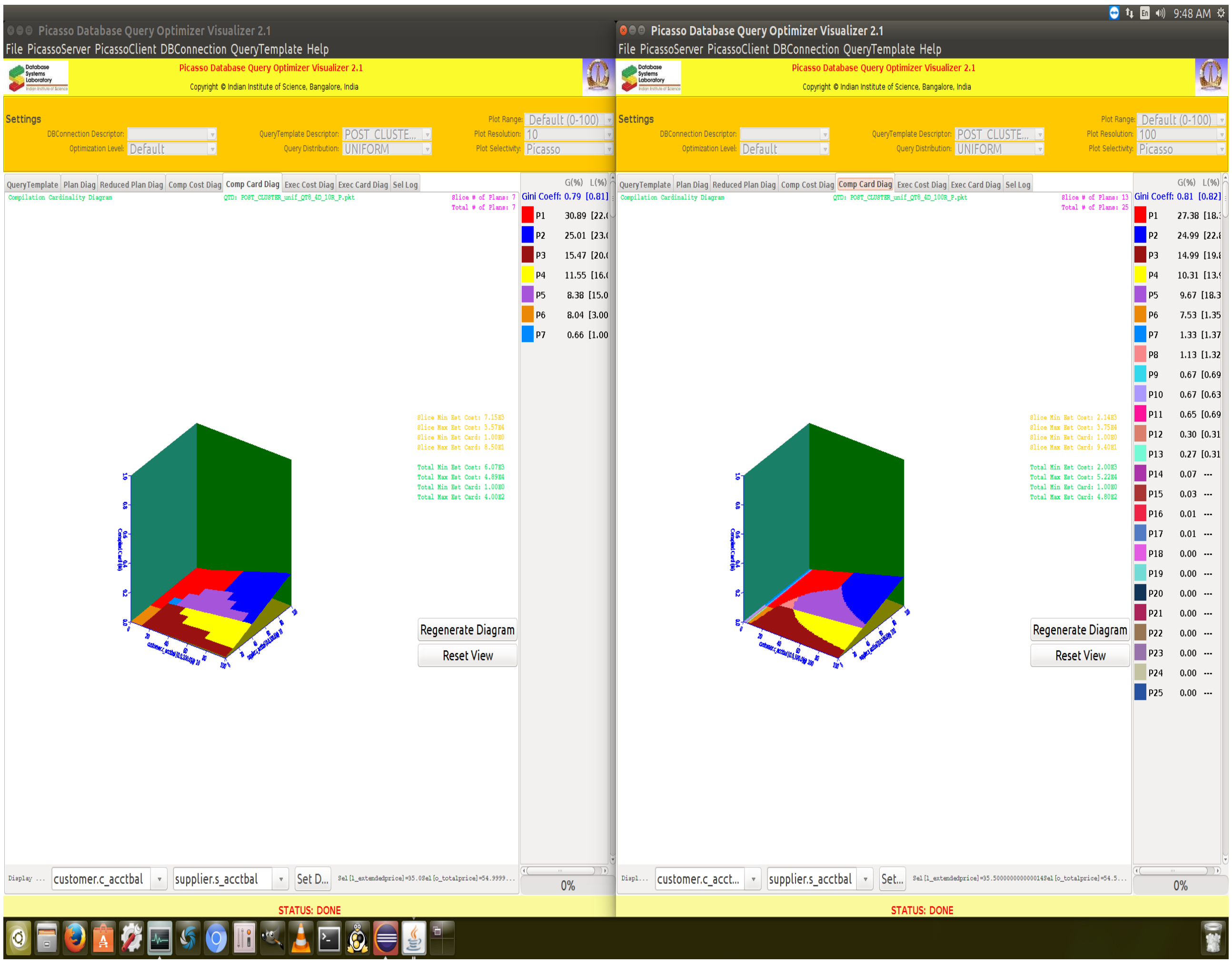
		G(%)	L(%)
Gini Coeff:		0.81	[0.82]
🟥	P1	27.38	[18.3
🟦	P2	24.99	[22.8
🟫	P3	14.99	[19.8
🟨	P4	10.31	[13.9
🟪	P5	9.67	[18.3
🟧	P6	7.53	[1.35
🟦	P7	1.33	[1.37
🟥	P8	1.13	[1.32
🟦	P9	0.67	[0.69
🟪	P10	0.67	[0.63
🟥	P11	0.65	[0.69
🟫	P12	0.30	[0.31
🟩	P13	0.27	[0.31
🟪	P14	0.07	---
🟫	P15	0.03	---
🟥	P16	0.01	---
🟦	P17	0.01	---
🟪	P18	0.00	---
🟦	P19	0.00	---
🟦	P20	0.00	---
🟥	P21	0.00	---
🟫	P22	0.00	---
🟪	P23	0.00	---
⬜	P24	0.00	---
🟦	P25	0.00	---

Displ... customer.c_acct... supplier.s_acctbal Set... Sel [l_extendedprice]=35.500000000000014Sel [o_totalprice]=54.5...

STATUS: DONE

File PicassoServer PicassoClient DBConnection QueryTemplate Help
QueryTemplate
Plan Diag
Reduced Plan Diag
Comp Cost Diag
Comp Card Diag
Exec Cost Diag
Exec Card Diag
Sel Log
Plan Diagram
supplier.s_acctbal [0.0,100.0]@ 10
customer.c_acctbal [0.0,100.0]@ 10
supplier.s_acctbal [0.0,100.0]@ 100
customer.c_acctbal [0.0,100.0]@ 100
Parameter → Operator Diff
Regenerate Diagram
Reset View
0%
9:46 AM

Figure 5.8: **Plan Diagram for o_t35&l_e54**



Picasso Database Query Optimizer Visualizer 2.1

Copyright © Indian Institute of Science, Bangalore, India

Settings

DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

Compilation Cardinality Diagram

QTD: POST_CLUSTER_unif_QT8_4D_10R_P.pkt

Slice # of Plans: 7
Total # of Plans: 7

Gini Coeff: 0.79 [0.81]

		G(%)	L(%)
🟥	P1	30.89	[22.(
🟦	P2	25.01	[23.(
🟥	P3	15.47	[20.(
🟨	P4	11.55	[16.(
🟪	P5	8.38	[15.0
🟧	P6	8.04	[3.00
🟦	P7	0.66	[1.00

```
Slice Min Est Cost: 7.15E3
Slice Max Est Cost: 3.57E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 8.50E1

Total Min Est Cost: 6.07E3
Total Max Est Cost: 4.89E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.00E2
```

Display ... customer.c_acctbal supplier.s_acctbal Set D... Sel [l_extendedprice]=35.0Sel [o_totalprice]=54.9999...

0%

STATUS: DONE

Picasso Database Query Optimizer Visualizer 2.1

Copyright © Indian Institute of Science, Bangalore, India

Settings

DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 100
Plot Selectivity: Picasso

Compilation Cardinality Diagram

QTD: POST_CLUSTER_unif_QT8_4D_100R_P.pkt

Slice # of Plans: 13
Total # of Plans: 25

Gini Coeff: 0.81 [0.82]

		G(%)	L(%)
🟥	P1	27.38	[18.:
🟦	P2	24.99	[22.8
🟥	P3	14.99	[19.8
🟨	P4	10.31	[13.9
🟪	P5	9.67	[18.3
🟧	P6	7.53	[1.35
🟦	P7	1.33	[1.37
🟥	P8	1.13	[1.32
🟦	P9	0.67	[0.69
🟪	P10	0.67	[0.63
🟥	P11	0.65	[0.69
🟫	P12	0.30	[0.31
🟩	P13	0.27	[0.31
🟪	P14	0.07	---
🟫	P15	0.03	---
🟥	P16	0.01	---
🟦	P17	0.01	---
🟪	P18	0.00	---
🟦	P19	0.00	---
🟦	P20	0.00	---
🟥	P21	0.00	---
🟫	P22	0.00	---
🟪	P23	0.00	---
🟫	P24	0.00	---
🟦	P25	0.00	---

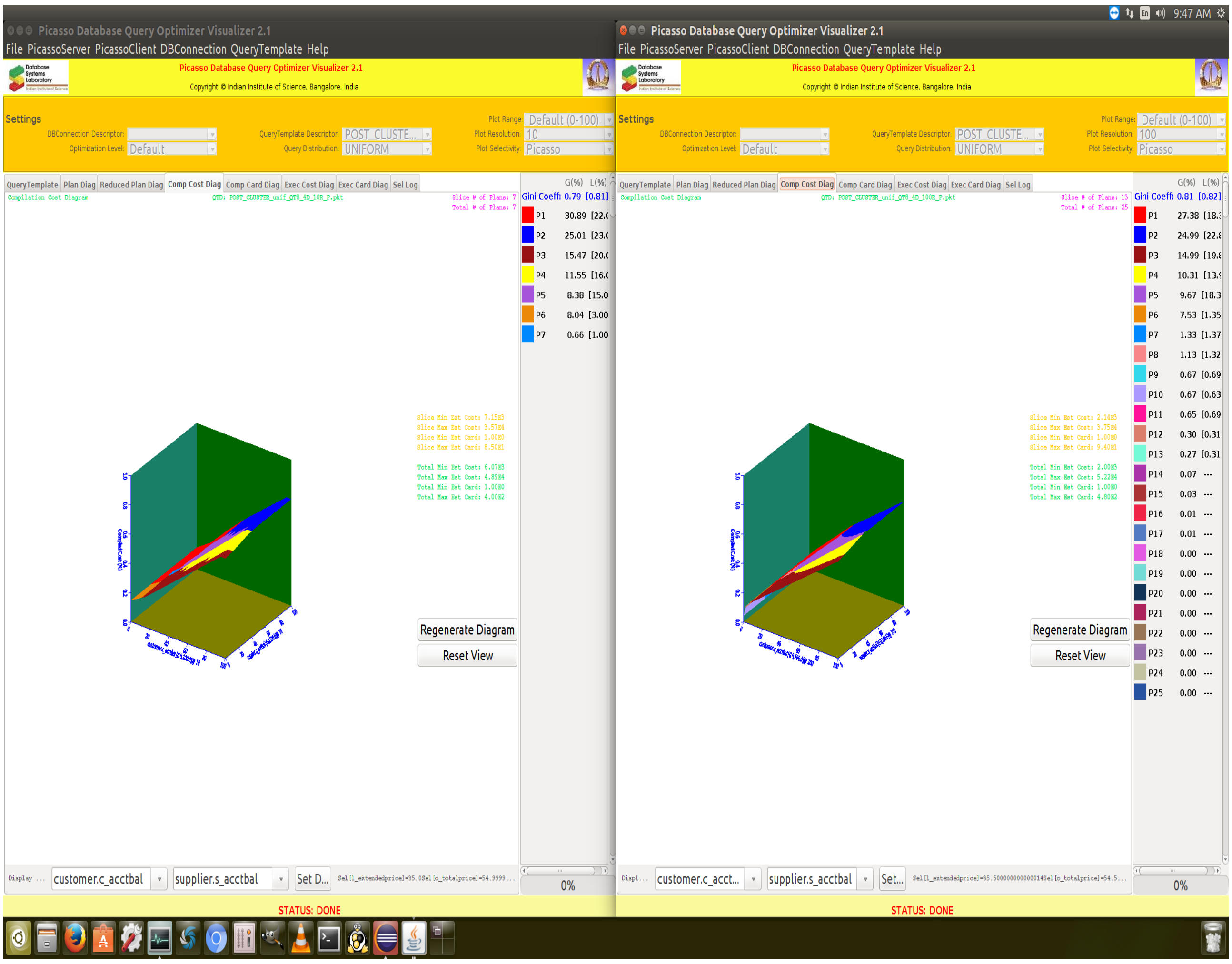
```
Slice Min Est Cost: 2.14E3
Slice Max Est Cost: 3.75E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 9.40E1

Total Min Est Cost: 2.00E3
Total Max Est Cost: 5.22E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.80E2
```

Displ... customer.c_acct... supplier.s_acctbal Set... Sel [l_extendedprice]=35.500000000000014Sel [o_totalprice]=54.5...

0%

STATUS: DONE

Figure 5.9: **Compilation Cardinality Diagram for o_t35&l_e54**


Picasso Database Query Optimizer Visualizer 2.1
File PicassoServer PicassoClient DBConnection QueryTemplate Help
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate Plan Diag Reduced Plan Diag Comp Cost Diag Comp Card Diag Exec Cost Diag Exec Card Diag Sel Log
Regenerate Diagram
Reset View
STATUS: DONE
9:47 AM

Settings

DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 10
Plot Selectivity: Picasso

Compilation Cost Diagram
QTD: POST_CLUSTER_unif_QT8_4D_10R_P.pkt
Slice # of Plans: 7
Total # of Plans: 7

```
Slice Min Est Cost: 7.15E3
Slice Max Est Cost: 3.57E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 8.50E1

Total Min Est Cost: 6.07E3
Total Max Est Cost: 4.89E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.00E2
```

Gini Coeff: 0.79 [0.81]

		G(%)	L(%)
🟥	P1	30.89	[22.(
🟦	P2	25.01	[23.(
🟫	P3	15.47	[20.(
🟨	P4	11.55	[16.(
🟪	P5	8.38	[15.0
🟧	P6	8.04	[3.00
🟦	P7	0.66	[1.00

Display ... customer.c_acctbal supplier.s_acctbal Set D... Sel [l_extendedprice]=35.0Sel [o_totalprice]=54.9999...

0%

Settings

DBConnection Descriptor:
Optimization Level: Default
QueryTemplate Descriptor: POST_CLUSTE...
Query Distribution: UNIFORM
Plot Range: Default (0-100)
Plot Resolution: 100
Plot Selectivity: Picasso

Compilation Cost Diagram
QTD: POST_CLUSTER_unif_QT8_4D_100R_P.pkt
Slice # of Plans: 13
Total # of Plans: 25

```
Slice Min Est Cost: 2.14E3
Slice Max Est Cost: 3.75E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 9.40E1

Total Min Est Cost: 2.00E3
Total Max Est Cost: 5.22E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.80E2
```

Gini Coeff: 0.81 [0.82]

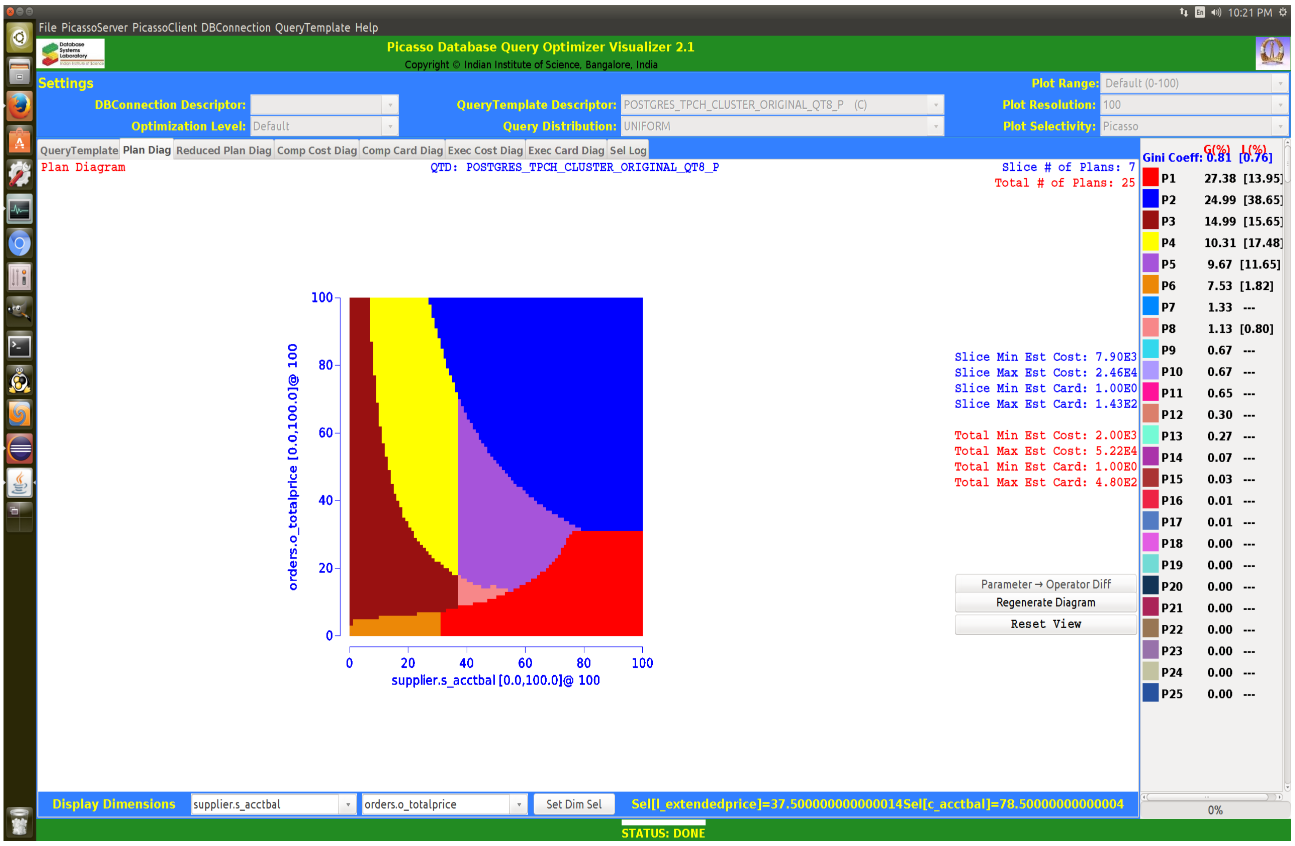
		G(%)	L(%)
🟥	P1	27.38	[18.:
🟦	P2	24.99	[22.8
🟫	P3	14.99	[19.8
🟨	P4	10.31	[13.9
🟪	P5	9.67	[18.3
🟧	P6	7.53	[1.35
🟦	P7	1.33	[1.37
🟥	P8	1.13	[1.32
🟦	P9	0.67	[0.69
🟪	P10	0.67	[0.63
🟥	P11	0.65	[0.69
🟫	P12	0.30	[0.31
🟩	P13	0.27	[0.31
🟪	P14	0.07	---
🟫	P15	0.03	---
🟥	P16	0.01	---
🟦	P17	0.01	---
🟪	P18	0.00	---
🟦	P19	0.00	---
🟦	P20	0.00	---
🟥	P21	0.00	---
🟫	P22	0.00	---
🟪	P23	0.00	---
⬜	P24	0.00	---
🟦	P25	0.00	---

Displ... customer.c_acct... supplier.s_acctbal Set... Sel [l_extendedprice]=35.500000000000014Sel [o_totalprice]=54.5...

0%

Figure 5.10: **Compilation Cost Diagram for o_t35&l_e54**

The following 3 figures show that the plan diagram, cardinality diagram and cost diagram are similar for the slices taken when the selectivities of *c_acctbal* is 78.5% and *l_extendedprice* is 37.5%


File PicassoServer PicassoClient DBConnection QueryTemplate Help
10:21 PM
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate
Plan Diag
Reduced Plan Diag
Comp Cost Diag
Comp Card Diag
Exec Cost Diag
Exec Card Diag
Sel Log
Plan Diagram
orders.o_totalprice [0.0,100.0]@ 100
supplier.s_acctbal [0.0,100.0]@ 100
Parameter → Operator Diff
Regenerate Diagram
Reset View
Display Dimensions
Set Dim Sel
STATUS: DONE

Settings

DBConnection Descriptor:

Optimization Level: Default

QueryTemplate Descriptor: POSTGRES_TPCH_CLUSTER_ORIGINAL_QT8_P (C)

Query Distribution: UNIFORM

Plot Range: Default (0-100)

Plot Resolution: 100

Plot Selectivity: Picasso

QTD: POSTGRES_TPCH_CLUSTER_ORIGINAL_QT8_P

Slice # of Plans: 7
Total # of Plans: 25

```
Slice Min Est Cost: 7.90E3
Slice Max Est Cost: 2.46E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 1.43E2

Total Min Est Cost: 2.00E3
Total Max Est Cost: 5.22E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.80E2
```

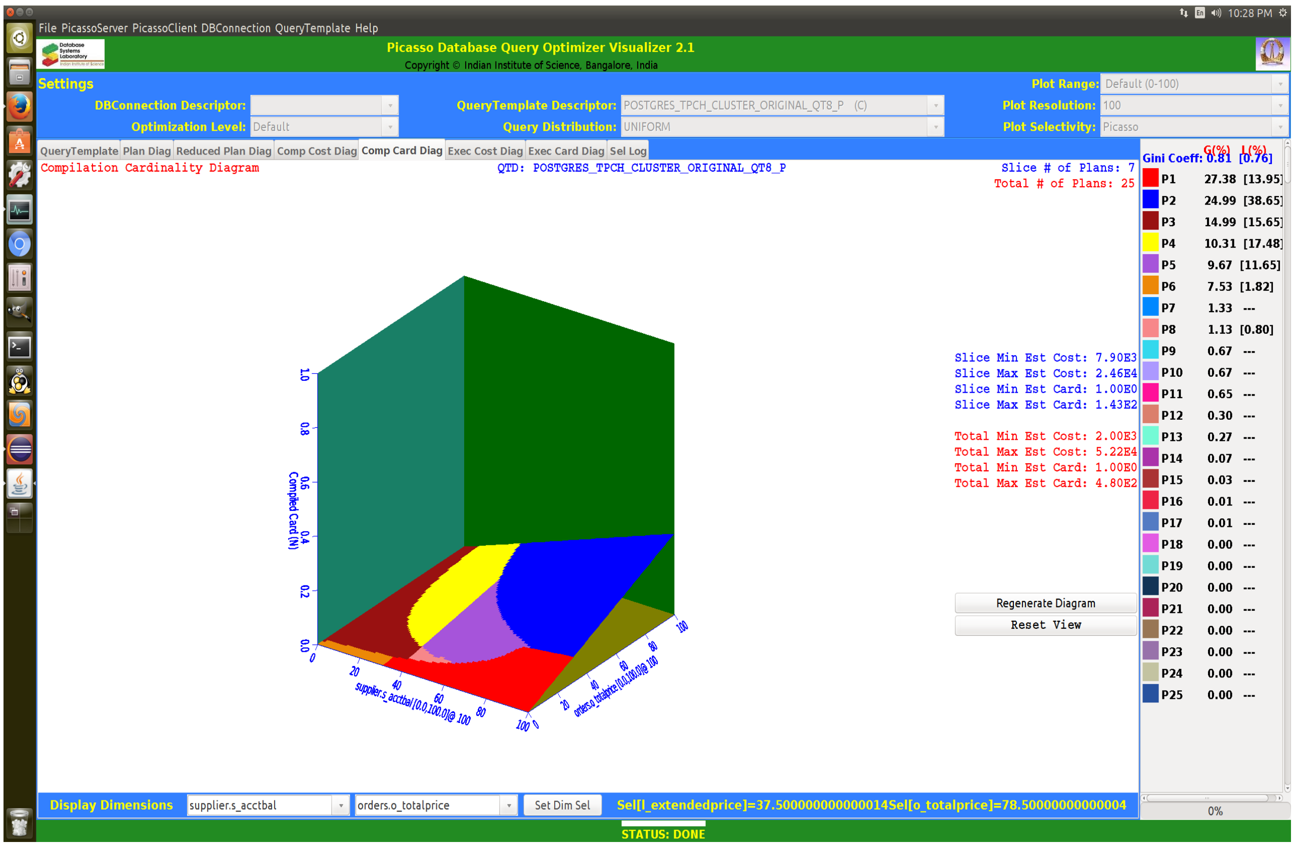
Display Dimensions: supplier.s_acctbal, orders.o_totalprice

Sel[l_extendedprice]=37.500000000000014Sel[c_acctbal]=78.50000000000004

	Gini Coeff:	G(%) 0.81	L(%) [0.76]
🟥	P1	27.38	[13.95]
🟦	P2	24.99	[38.65]
🟥	P3	14.99	[15.65]
🟨	P4	10.31	[17.48]
🟪	P5	9.67	[11.65]
🟧	P6	7.53	[1.82]
🟦	P7	1.33	---
🟥	P8	1.13	[0.80]
🟦	P9	0.67	---
🟪	P10	0.67	---
🟪	P11	0.65	---
🟫	P12	0.30	---
🟩	P13	0.27	---
🟪	P14	0.07	---
🟫	P15	0.03	---
🟥	P16	0.01	---
🟦	P17	0.01	---
🟪	P18	0.00	---
🟩	P19	0.00	---
🟦	P20	0.00	---
🟥	P21	0.00	---
🟫	P22	0.00	---
🟪	P23	0.00	---
⬜	P24	0.00	---
🟦	P25	0.00	---

0%

Figure 5.11: **Plan Diagram for l_e37.5&c_a78.5**



Picasso Database Query Optimizer Visualizer 2.1

Copyright © Indian Institute of Science, Bangalore, India

Settings

DBConnection Descriptor:

Optimization Level: Default

QueryTemplate Descriptor: POSTGRES_TPCH_CLUSTER_ORIGINAL_QT8_P (C)

Query Distribution: UNIFORM

Plot Range: Default (0-100)

Plot Resolution: 100

Plot Selectivity: Picasso

Compilation Cardinality Diagram

QTD: POSTGRES_TPCH_CLUSTER_ORIGINAL_QT8_P

Slice # of Plans: 7
Total # of Plans: 25

```
Slice Min Est Cost: 7.90E3
Slice Max Est Cost: 2.46E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 1.43E2

Total Min Est Cost: 2.00E3
Total Max Est Cost: 5.22E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.80E2
```

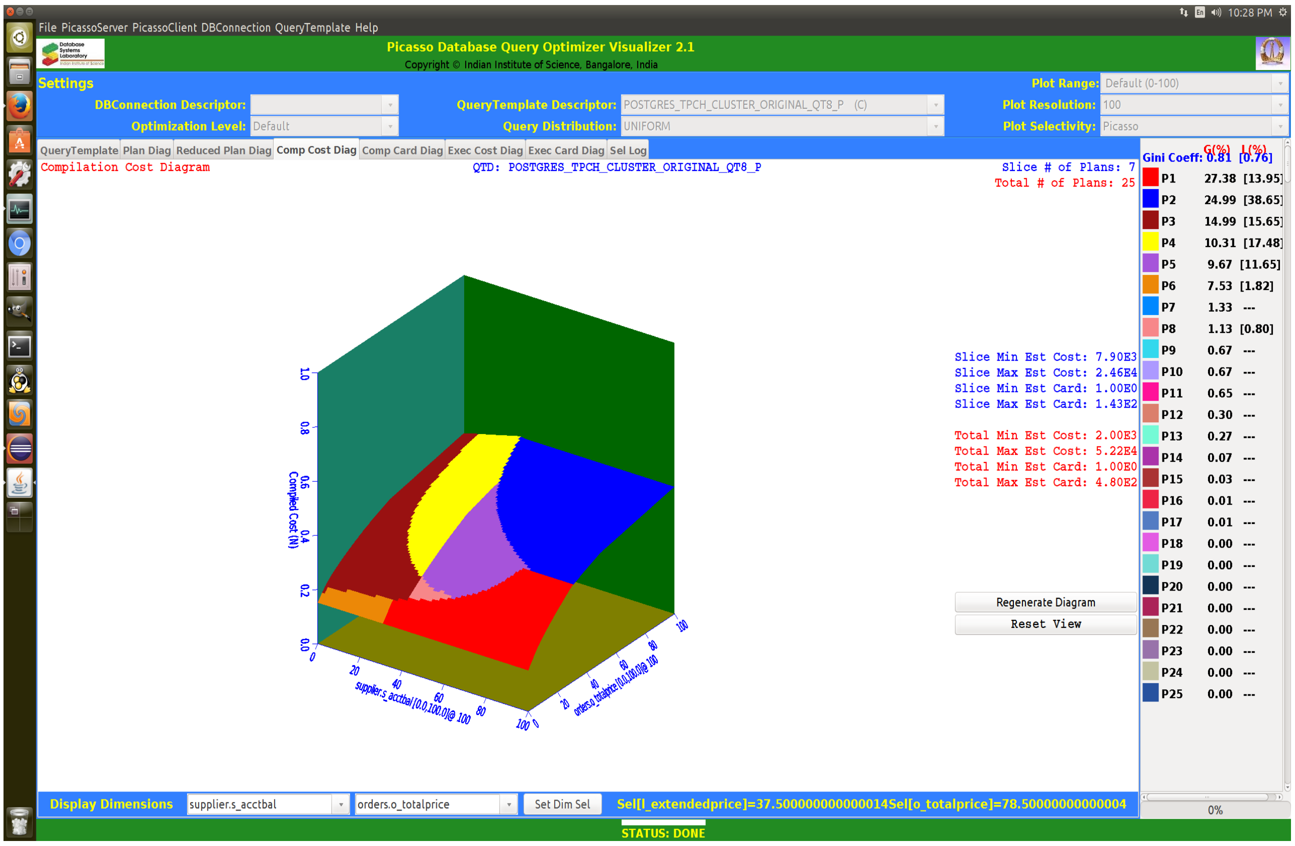
Gini Coeff:		0.81	[0.76]
		G(%)	L(%)
🟥	P1	27.38	[13.95]
🟦	P2	24.99	[38.65]
🟫	P3	14.99	[15.65]
🟨	P4	10.31	[17.48]
🟪	P5	9.67	[11.65]
🟧	P6	7.53	[1.82]
🟦	P7	1.33	---
🟥	P8	1.13	[0.80]
🟦	P9	0.67	---
🟪	P10	0.67	---
🟥	P11	0.65	---
🟫	P12	0.30	---
🟩	P13	0.27	---
🟪	P14	0.07	---
🟫	P15	0.03	---
🟥	P16	0.01	---
🟦	P17	0.01	---
🟪	P18	0.00	---
🟩	P19	0.00	---
⬛	P20	0.00	---
🟥	P21	0.00	---
🟫	P22	0.00	---
🟪	P23	0.00	---
⬜	P24	0.00	---
🟦	P25	0.00	---

0%

Display Dimensions supplier.s_acctbal orders.o_totalprice

Sel[l_extendedprice]=37.500000000000014Sel[o_totalprice]=78.50000000000004

Figure 5.12: **Compilation Cardinality Diagram for l_e37.5&c_a78.5**


File PicassoServer PicassoClient DBConnection QueryTemplate Help
10:28 PM
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate
Plan Diag
Reduced Plan Diag
Comp Cost Diag
Comp Card Diag
Exec Cost Diag
Exec Card Diag
Sel Log
Compiled Cost (N)
0.0
0.2
0.4
0.6
0.8
1.0
supplier.s_acctbal [0.0,100.0]@ 100
orders.o_totalprice [0.0,100.0]@ 100
0
20
40
60
80
100
Regenerate Diagram
Reset View
Set Dim Sel
STATUS: DONE

Settings

DBConnection Descriptor:

Optimization Level: Default

QueryTemplate Descriptor: POSTGRES_TPCH_CLUSTER_ORIGINAL_QT8_P (C)

Query Distribution: UNIFORM

Plot Range: Default (0-100)

Plot Resolution: 100

Plot Selectivity: Picasso

Compilation Cost Diagram

QTD: POSTGRES_TPCH_CLUSTER_ORIGINAL_QT8_P

Slice # of Plans: 7
Total # of Plans: 25

```
Slice Min Est Cost: 7.90E3
Slice Max Est Cost: 2.46E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 1.43E2

Total Min Est Cost: 2.00E3
Total Max Est Cost: 5.22E4
Total Min Est Card: 1.00E0
Total Max Est Card: 4.80E2
```

Gini Coeff:		G(%) 0.81	L(%) [0.76]
🟥	P1	27.38	[13.95]
🟦	P2	24.99	[38.65]
🟥	P3	14.99	[15.65]
🟨	P4	10.31	[17.48]
🟪	P5	9.67	[11.65]
🟧	P6	7.53	[1.82]
🟦	P7	1.33	---
🟥	P8	1.13	[0.80]
🟦	P9	0.67	---
🟪	P10	0.67	---
🟥	P11	0.65	---
🟥	P12	0.30	---
🟩	P13	0.27	---
🟪	P14	0.07	---
🟥	P15	0.03	---
🟥	P16	0.01	---
🟦	P17	0.01	---
🟪	P18	0.00	---
🟦	P19	0.00	---
🟦	P20	0.00	---
🟥	P21	0.00	---
🟫	P22	0.00	---
🟪	P23	0.00	---
⬜	P24	0.00	---
🟦	P25	0.00	---

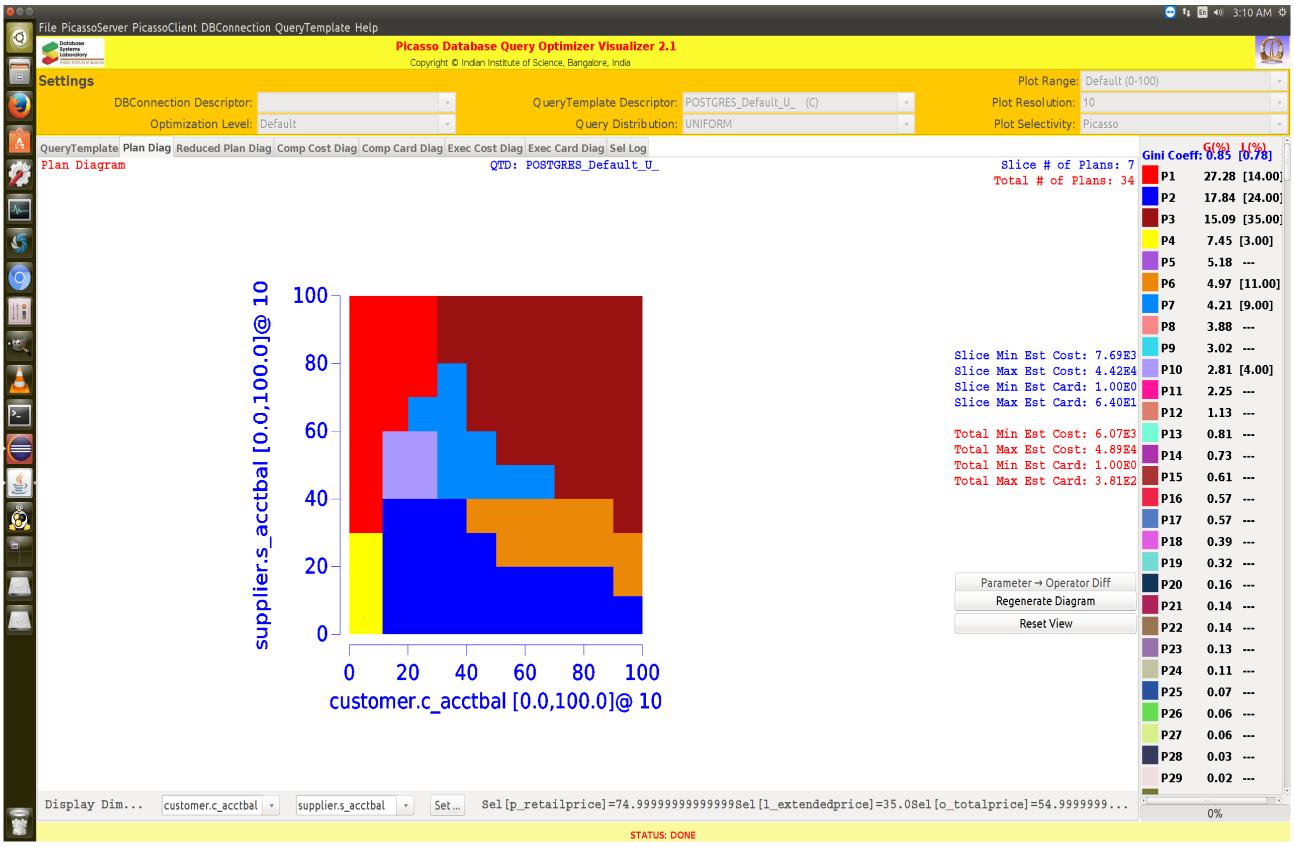
Display Dimensions supplier.s_acctbal orders.o_totalprice

Sel[l_extendedprice]=37.500000000000014Sel[o_totalprice]=78.50000000000004

0%

Figure 5.13: **Compilation Cost Diagram for l_e37.5&c_a78.5**

The following figure shows a slice of a plan diagram for 5 dimensions where *p_retailprice* is 75% and *l_extendedprice* is 35% and *o_totalprice* is 55% on adding *p_retailprice* as 5th error-prone predicate in the TPC-H Q8 query.



File PicassoServer PicassoClient DBConnection QueryTemplate Help
3:10 AM
Picasso Database Query Optimizer Visualizer 2.1
Copyright © Indian Institute of Science, Bangalore, India
QueryTemplate
Plan Diag
Reduced Plan Diag
Comp Cost Diag
Comp Card Diag
Exec Cost Diag
Exec Card Diag
Sel Log
Plan Diagram
supplier.s_acctbal [0.0,100.0]@ 10
customer.c_acctbal [0.0,100.0]@ 10
100
80
60
40
20
0
0
20
40
60
80
100
Parameter → Operator Diff
Regenerate Diagram
Reset View
Display Dim...
Set ...
STATUS: DONE

Settings

DBConnection Descriptor:

Optimization Level: Default

QueryTemplate Descriptor: POSTGRES_Default_U_ (C)

Query Distribution: UNIFORM

Plot Range: Default (0-100)

Plot Resolution: 10

Plot Selectivity: Picasso

QTD: POSTGRES_Default_U_

Slice # of Plans: 7
Total # of Plans: 34

```
Slice Min Est Cost: 7.69E3
Slice Max Est Cost: 4.42E4
Slice Min Est Card: 1.00E0
Slice Max Est Card: 6.40E1

Total Min Est Cost: 6.07E3
Total Max Est Cost: 4.89E4
Total Min Est Card: 1.00E0
Total Max Est Card: 3.81E2
```

Gini Coeff: 0.85 [0.78]

		G(%)	L(%)
🟥	P1	27.28	[14.00]
🟦	P2	17.84	[24.00]
🟫	P3	15.09	[35.00]
🟨	P4	7.45	[3.00]
🟪	P5	5.18	---
🟧	P6	4.97	[11.00]
🟦	P7	4.21	[9.00]
🟥	P8	3.88	---
🟦	P9	3.02	---
🟪	P10	2.81	[4.00]
🟥	P11	2.25	---
🟫	P12	1.13	---
🟩	P13	0.81	---
🟪	P14	0.73	---
🟫	P15	0.61	---
🟥	P16	0.57	---
🟦	P17	0.57	---
🟪	P18	0.39	---
🟦	P19	0.32	---
🟦	P20	0.16	---
🟥	P21	0.14	---
🟫	P22	0.14	---
🟪	P23	0.13	---
⬜	P24	0.11	---
🟦	P25	0.07	---
🟩	P26	0.06	---
🟩	P27	0.06	---
⬛	P28	0.03	---
⬜	P29	0.02	---

customer.c_acctbal | supplier.s_acctbal

Sel [p_retailprice] =74.99999999999999Sel [l_extendedprice] =35.0Sel [o_totalprice] =54.9999999...

0%

Figure 5.14: **Plan Diagram for p_75&l_35&o_55**

Hence, we have verified that the parallelization of POSP is giving the correct results and we were able to develop Parallelized PICASSO by creating picasso packets from these parallelized results.

Chapter 6

Conclusion and Future Work

In the initial work, we have automated the parallelization process to generate POSP on multi-core clusters and extended it to queries having error-prone join predicates and saved the results in binary format so that they occupy less storage space. In our second contribution, we have developed Parallelized PICASSO which operates in two phases. First being "Robust Query Processing" [5] which does generation of POSP on multi-core clusters by doing parallelization using MPI and C++. Second being the generation of Picasso Packet from these results implemented in JAVA. In our third contribution, we have modified the existing PICASSO to store the picasso packets in lesser storage space without loosing any information.

In our future work, we use Foreign Plan Cost (FPC) feature which tells us the behaviour of a particular plan over the entire selectivity space. In "Plan Bouquet" [3] and "Platform-independent Robust Query Processing" [7], experiments were carried out for queries having high dimensions (6) but low resolution (10). We will carry out the same experiments for high dimensions (4 and 5) and higher resolution (100) to verify the Maximum Sub-Optimality (MSO) guarantees given by them.

Bibliography

[1] http://www.serc.iisc.in/facilities/delta-cluster/. 12

[2] http://www.tpc.org/tpch/. 12

[3] Anshuman Dutt and Jayant R Haritsa. *Plan Bouquets: Query Processing without Selectivity Estimation*. In *SIGMOD*, 2014. ii, 1, 17, 28

[4] Anshuman Dutt, Sumit Neelam, and Jayant R Haritsa. *Quest: An Exploratory Approach to Robust Query Processing*. In *PVLDB*, 2014. 1, 17

[5] Kuntal Ghosh. *Robust Query Processing*. Master's thesis, Dept. of Computer Science and Automation, IISc, Bangalore, 2016. ii, 1, 2, 3, 4, 8, 28

[6] Jayant R Haritsa. *The Picasso Database Query Optimizer Visualizer*. In *VLDB*, 2010. ii, 2, 5, 6

[7] Srinivas Karthik, Jayant R Haritsa, Sreyash Kenkre, and Vinayaka Pandit. *Platform-independent robust query processing*. In *ICDE*, 2016. 17, 28