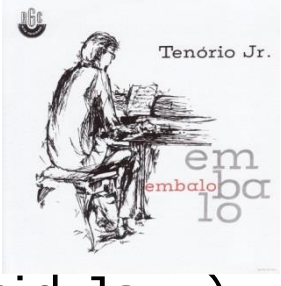
Who am I?

- **Tatsuro Yamada**
  - From Tokyo Japan
  - Work for NTT Open Source Software center

- **Work**
  - Database consulting and support for NTT Group companies
  - Performance evaluation: ex. PG95 on HP Superdome X (240core)
  - Just started contributing to oracle_fdw



- **Interest**
  - Listening to Music (Bossa nova, Jazz samba and Acid Jazz)
  - Skiing, Beer, RaspberryPi3, Android, Autocross

A Challenge of Huge Billing System Migration

Tatsuro Yamada
NTT OSS Center

PGCon May 20, 2016

Purpose of the presentation

Innovative R&D by NTT

- In this talk, I am going introduce barriers and solutions for our migration project.
- I hope to prove that PostgreSQL can be used in mission-critical fields.
- I hope to increase PostgreSQL users and they will share new use cases.

NTT

Agenda

1. Background of the migration project

2. Three major challenges of the project

3. Thoughts about the future of PostgreSQL

4. Conclusion

1. Background of the migration project

About NTT OSS Center

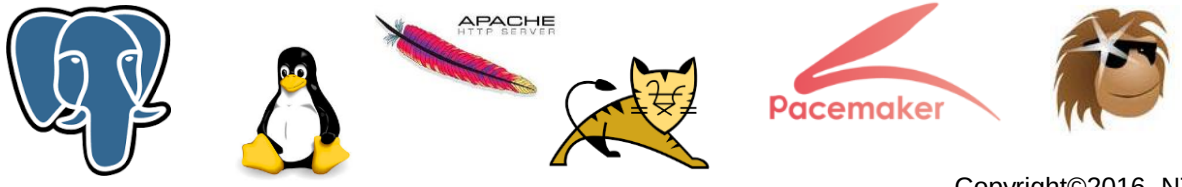
- **Who are we?**
  - NTT (Nippon Telegraph and Telephone Corporation)
  - Telecommunication Carrier in Japan
  - 900 subsidiaries throughout the world

- **What is NTT OSS Center?**
  - promotes and adopts OSS to the NTT group companies to reduce the TCO.
    - Consulting/support

      consulting service, support desk, product maintenance.
    - R&D

      developing OSS and related tools with the community.

      pg_statsinfo, pg_hint_plan, pg_bulkload, etc.



Relevant companies

- The Client, the SI and NTT OSS center are cooperation.
- We developed the migration system that utilizes the OSS.

Client

NTT Communications

- VPN services
- Mobile services
- Cloud services
- IP Telephone services and so on.

Cooperation

Systems Integrator

NTT Comware

Development Project

- Development
- Operation, maintenance

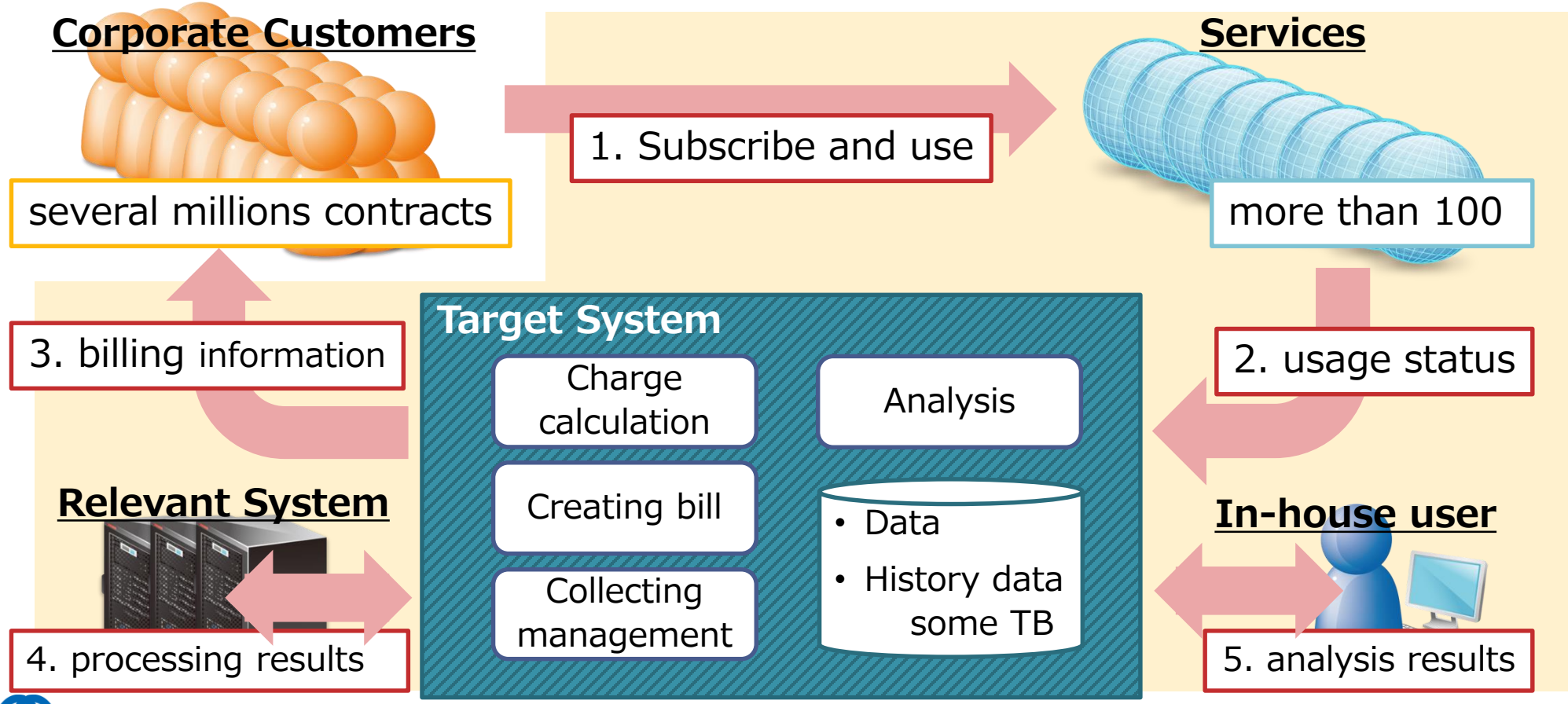
Consultant

NTT OSS Center

- Consulting/Support service of OSS

System Outline

- **The backbone billing system of a telecom business which has several millions of customers.**
- **Never allowed an unexpected shutdown nor even an error in life time: Mission-critical system.**



Why did we need system migration?

Innovative R&D by NTT

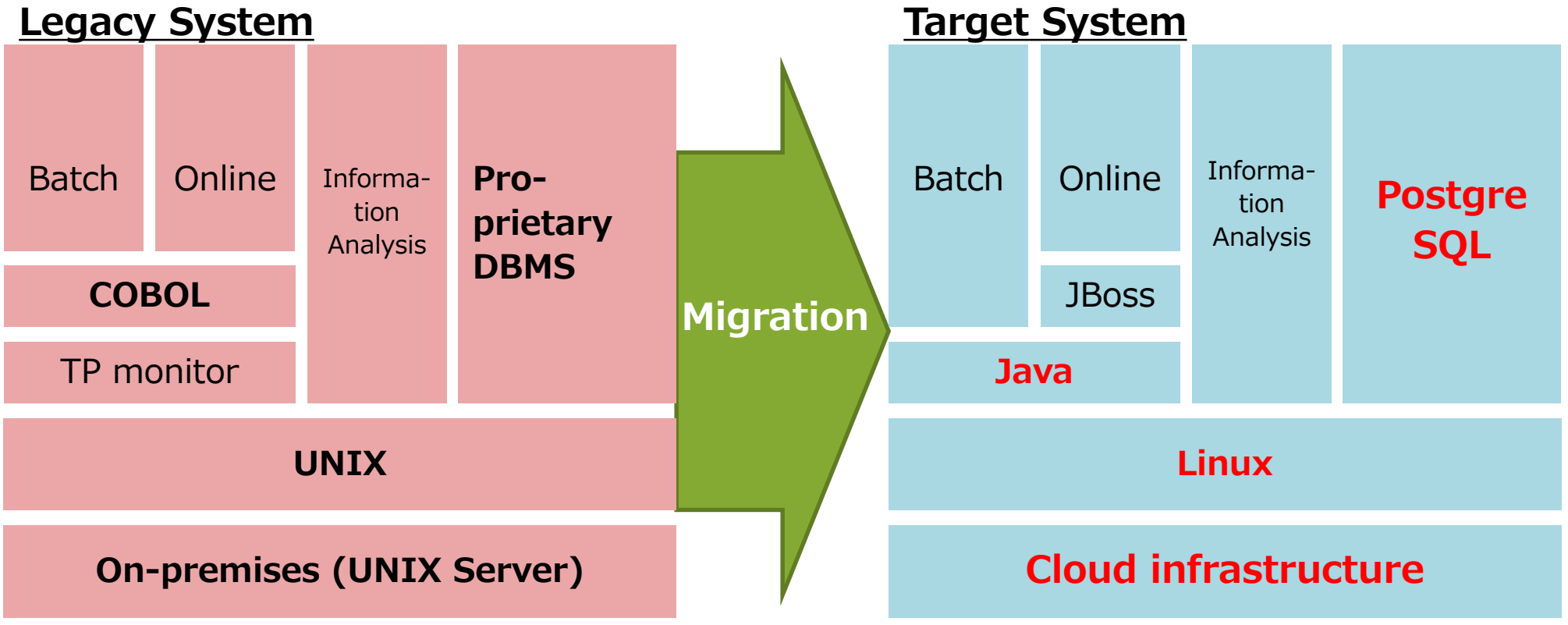
- **3 reasons for migration on the system**
  - **Life time**
    The legacy system approached end of life.
  - **Cost reduction**
    Discard proprietary software for cost reduction.
    ex. OS, DBMS
    - Also avoiding vendor lock-ins.
  - **Performance Scalability**
    Use cloud infrastructure to prepare for future business expansion.
    - To adopt new architecture: ex. Java instead of COBOL
- **We decided to rebuild the system, including OS, DBMS, application, table schemas and language.**

NTT

Comparison of legacy vs. target system

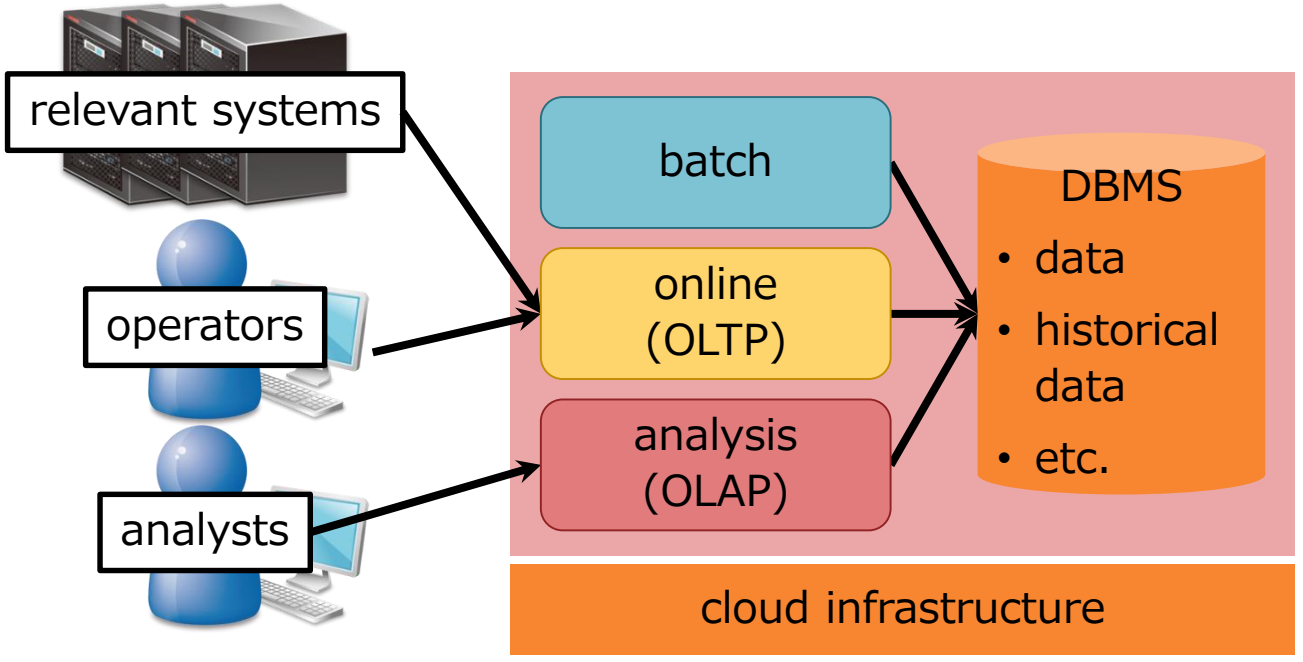
Innovative R&D by NTT

- **Software stack diagram**



System requirements

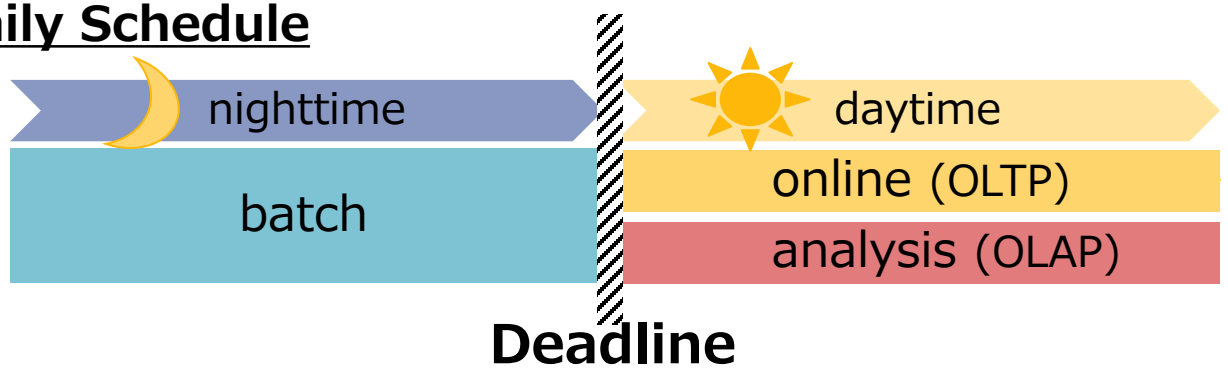
System Image



requirements

- 24x7, mission critical
- batch, online and analysis
- 5000 batch jobs/day
- data size is over 1TB
- **have to use cloud**

Daily Schedule

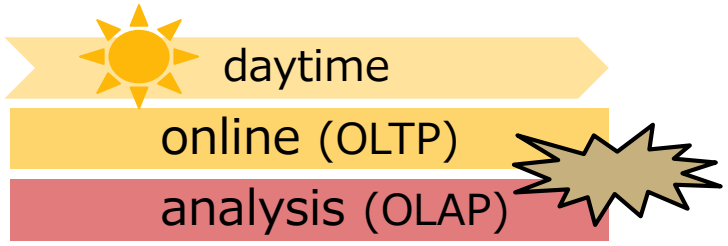


- **strict batch performance requirement and time correctness**
- **OLTP and OLAP work together during the day**

3. Project Challenges

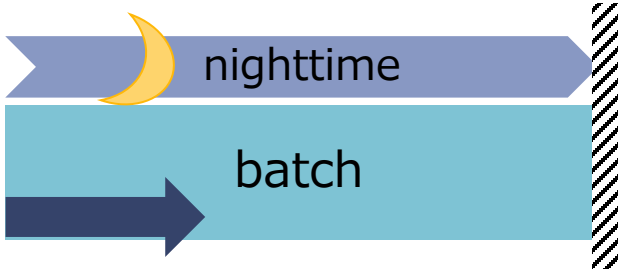
Three major challenges in the project

1. To run OLTP and OLAP without interfering each other



2. To guarantee performance on cloud

3. To ensure batch performance stability until EOL



First challenge

1. To run OLTP and OLAP without interfering each other

2. To guarantee performance on cloud

3. To ensure batch performance stability until EOL

OLTP and OLAP without interfering

- **Challenge**
  - Real-time analysis
  - No negative effect on OLTP performance
- **Our solution**
  - **Divided OLTP and OLAP into separate databases**
  - Used Stream Replication with this setting:
    - **eliminate query cancels** caused by WAL replay on the read replica.

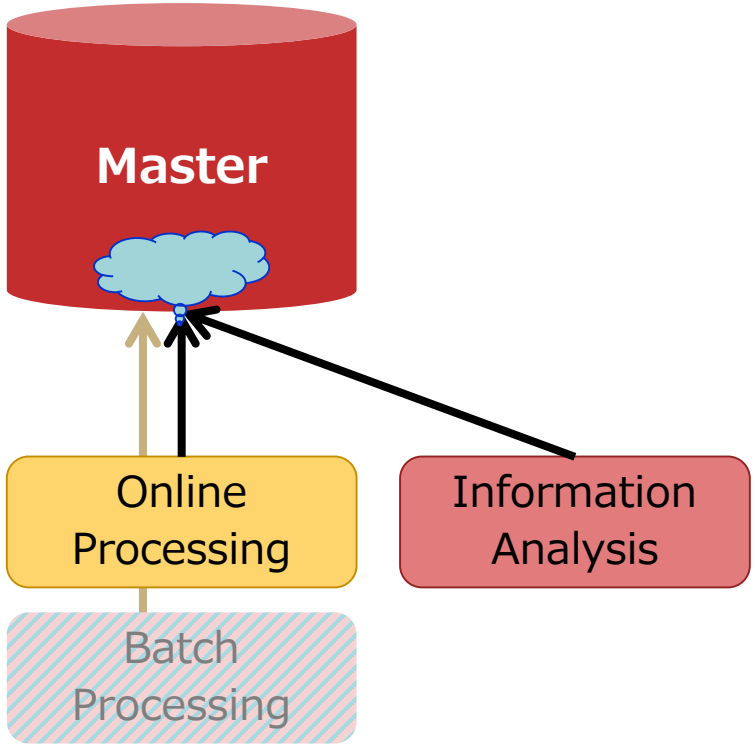
      max_standby_streaming_delay = -1

NTT

Why did we decide it?

Innovative R&D by NTT

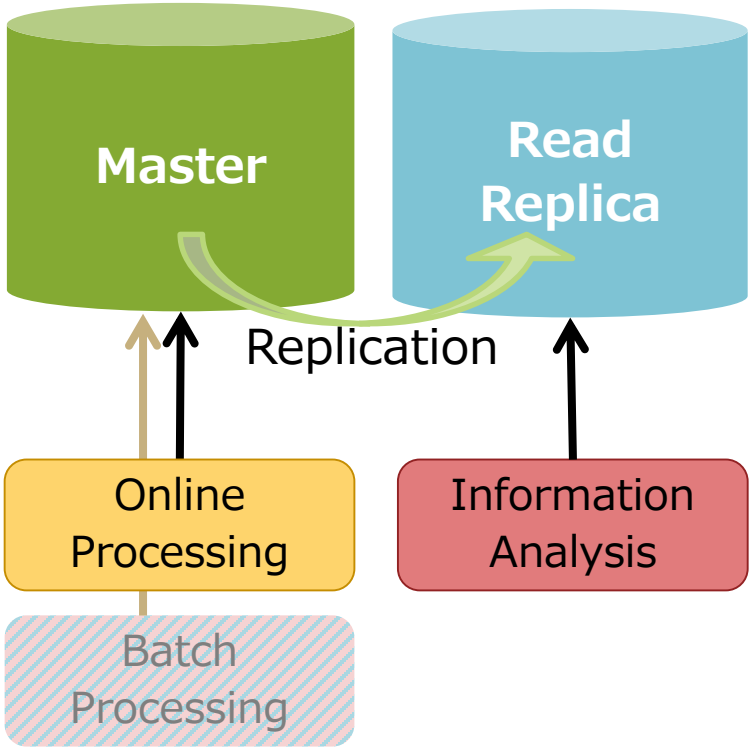
Legacy Design



- OLTP and OLAP scrambles for resources, brings to negative effect to each other.

Bad

Our Design



- OLTP and OLAP can use own resources, There is no effect to each other.

Good

NTT

Problem happened!

- **We thought it would be easy to create and run a read replica, but we ran into a few problems.**

- **Problem**
  - WAL replay suddenly stopped on the read replica.
    - No real-time analysis (query results staled)

Investigation of the problem

- **Cause?**
  - Many long queries, WAL replay was waiting for finished query execution on the read replica.
  - There is a side effect,
    In the case of max_standby_streaming_delay = -1.
- **Solution**
  - add hot_standby_feedback = ON to the settings.
    - the Master database does not send a WAL which causes conflict against the running query on the read-replica.
- **The complete solution for OLAP is:**
  - **max_standby_streaming_delay = -1**
  - **hot_standby_feedback = ON**

Challenges

1. To run OLTP and OLAP without interfering each other

2. To guarantee performance on cloud

3. To ensure batch performance stability until EOL

Second Challenge

1. To run OLTP and OLAP without affecting each other

2. To guarantee performance on the cloud

3. To ensure batch performance stability until EOL

To guarantee performance on the cloud

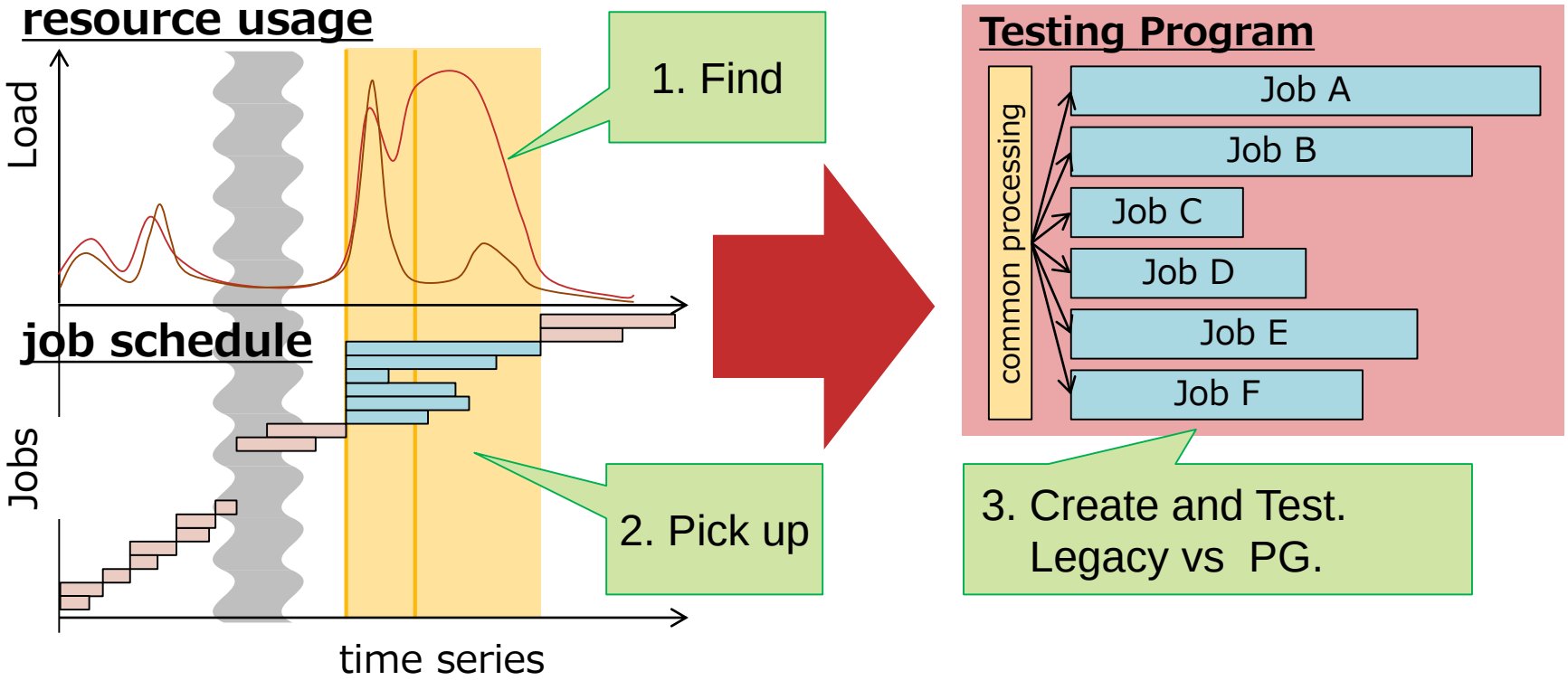
- **Challenge**
  - Usually resources(I/O, CPU and memory) are not guaranteed on a cloud environment
  - **Need to finish batch jobs before deadline**
- **What should we do then?**

What should we do?

- **Our Solution: Performance evaluation**

  Steps

  1. Find the highest time zone of the resource usage
  2. Pick out the specific jobs from the job schedule
  3. Create a testing program and measure the performance.



(*) Job: One unit of programs that make up the batch processing.
It is composed of a large number of SQL.

Results of performance evaluation

- **Results**
  - I/O usage was larger than expected.
    - We recognized the necessary number of I/O (IOPS).
  - We were able to tell required IOPS to the cloud vendor.
  - They promised us to guarantee the IOPS.
  - **We could manage to finish batch processing before deadline on the cloud.**

Challenges

1. To run OLTP and OLAP without interfering each other

2. **To guarantee performance on cloud**

3. To ensure batch performance stability until EOL

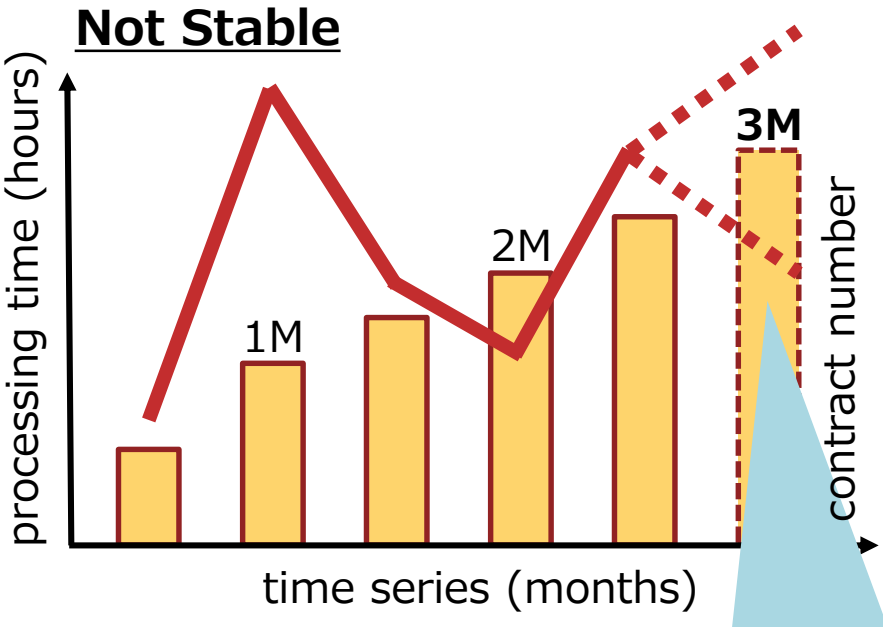
Third Challenge

1. To run OLTP and OLAP without affecting each other ✓
2. To guarantee performance on cloud ✓
3. **To ensure batch performance stability until EOL**

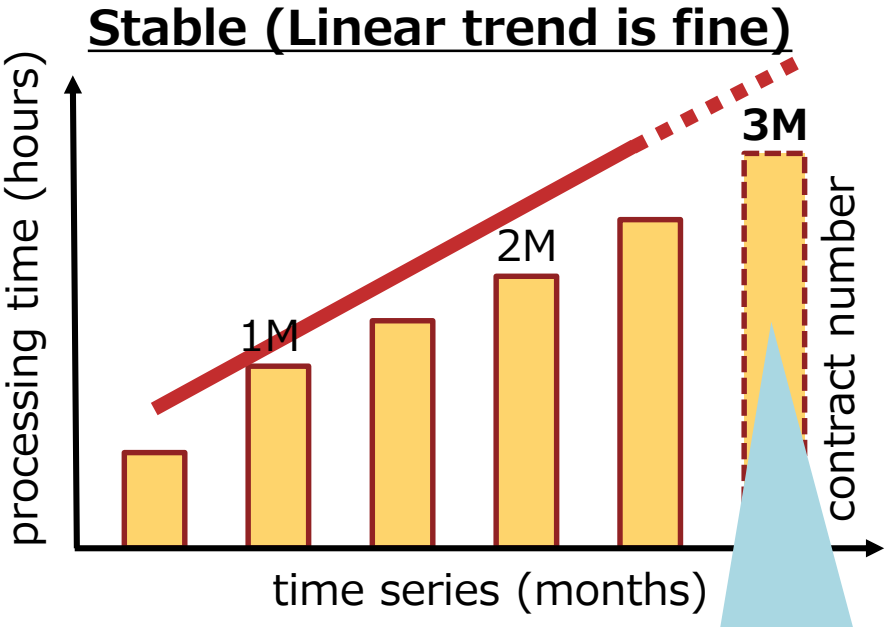
Why is batch performance stability needed?

- **Want to know the timing to enhance servers.**
  - The batch time window can't extend.
  - The linear trend is ideal, It's' easy to predict future batch performance.
  - able to add extra resources before running out.

Examples



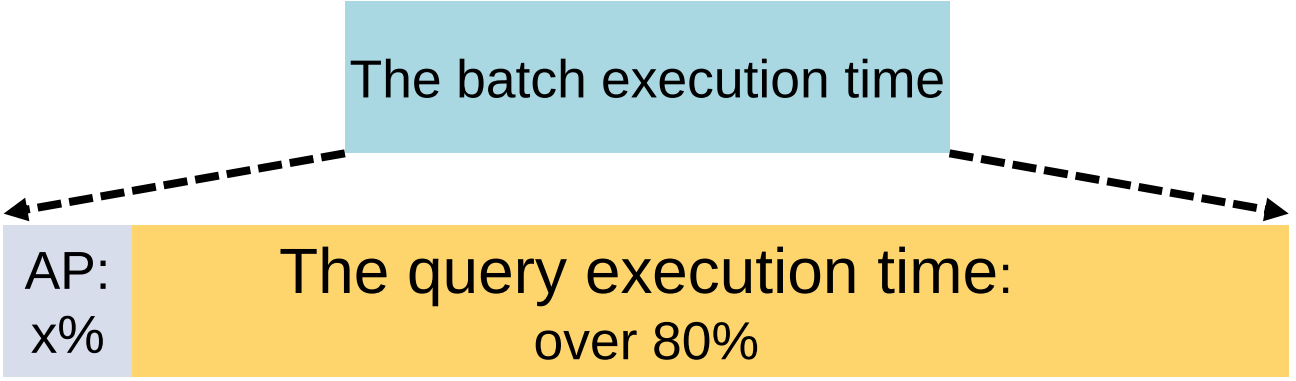
Difficult to predict the future processing time



Easy to predict it

To ensure batch performance stability

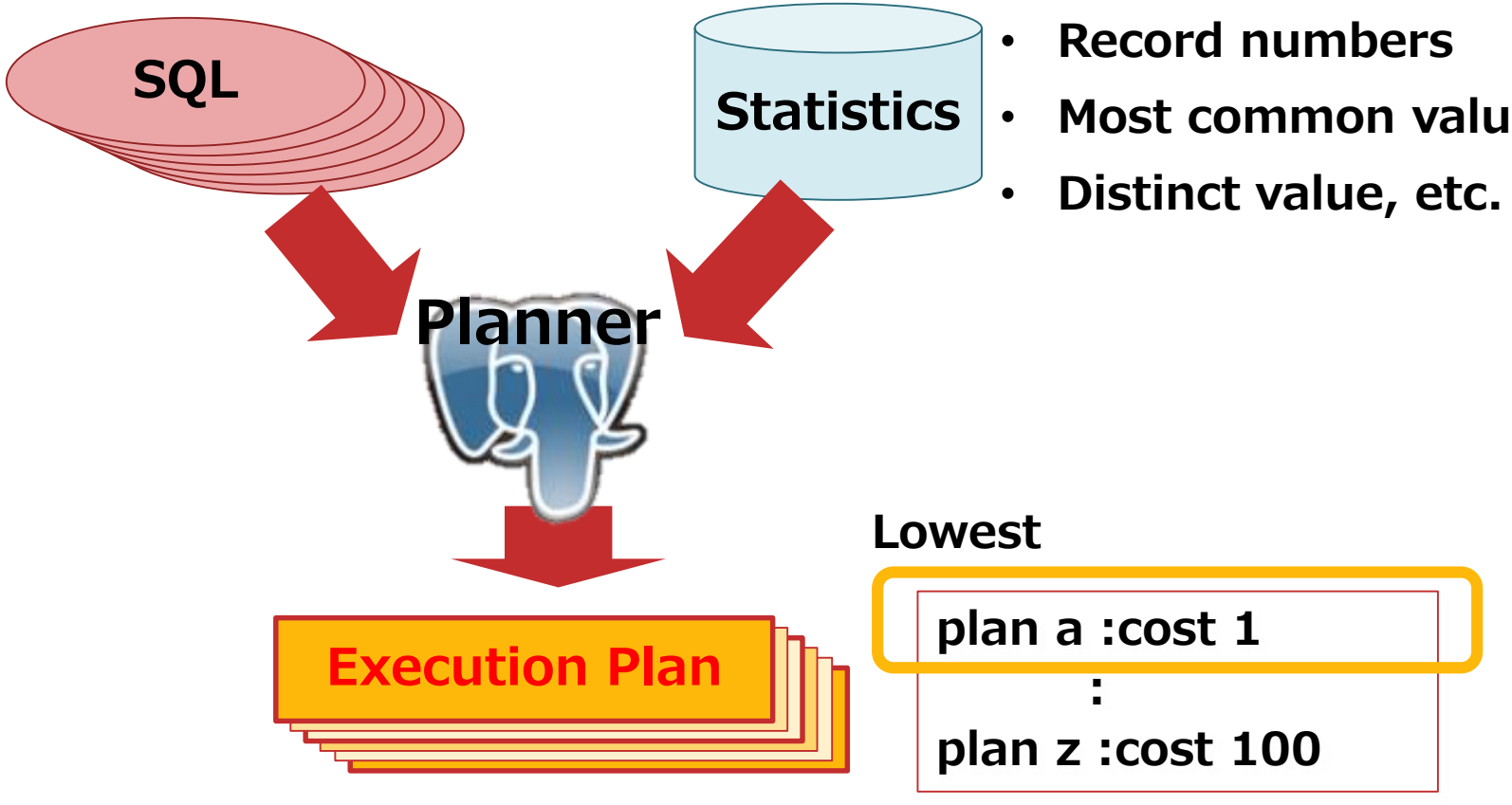
- **Analyze the batch jobs**
  - The batch execution time is almost equal to the query execution time on the system.



- **We Focus on the query execution time.**
- **What does “Query execution time” depend on?**
  - **The execution plan**.

Basics of query execution

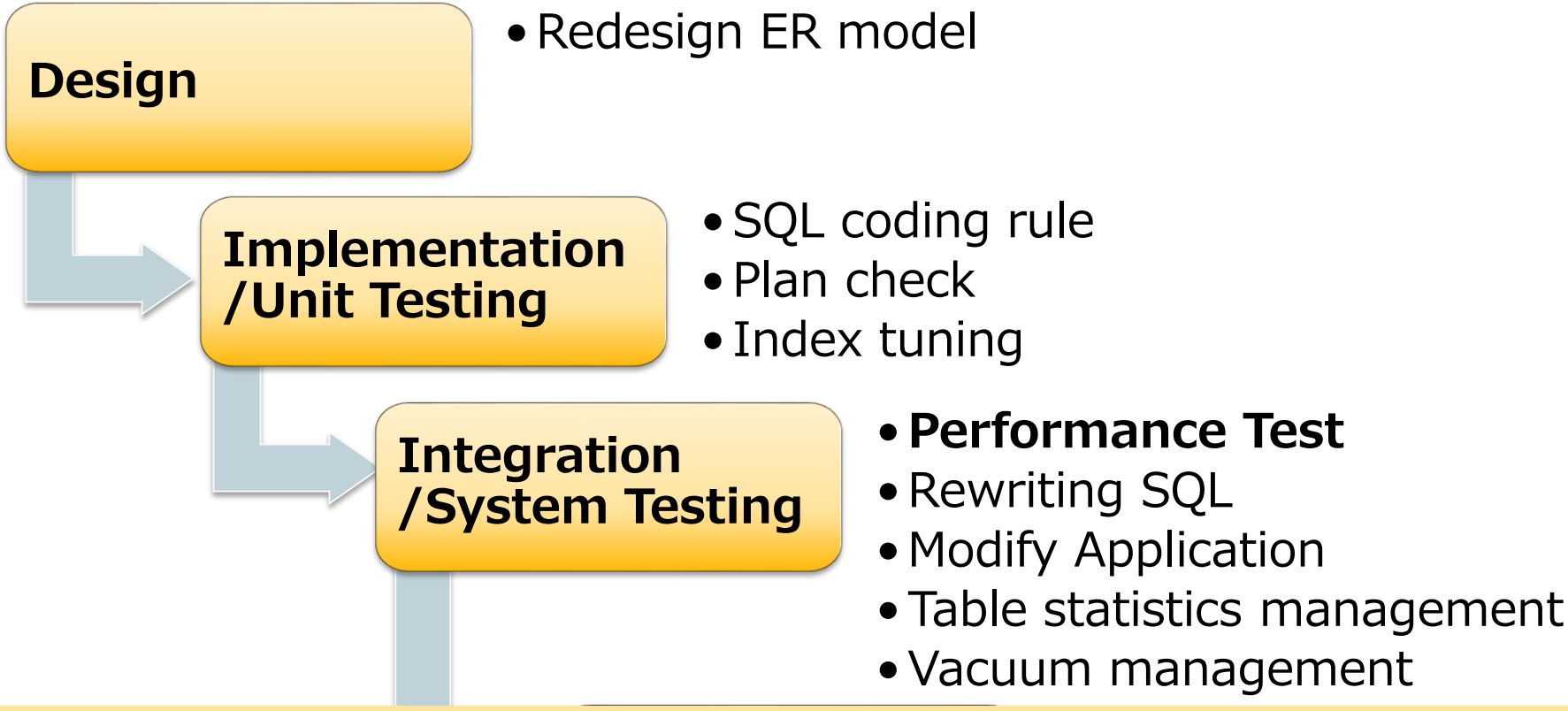
- How was the plan created and chosen?



- The planner makes a mistake sometimes.

What has been done to get a efficient plan?

- **We did these in each phase.**



The majority of queries met the requirements and were stable.

What happened?

- **Problem**
  - By changing a values of search criteria, query execution time increased from **several minutes to several days!**
    - The table record had not been increased.
    - The cardinality of the value was confirmed to be unchanged.
- **Cause?**
  - **The execution plan is inefficient.**
    The problem came from the planner.

The decision

- **If you were us, which would you choose for the solution?**
  - The premise:
    - worked variously to get the efficient plan.
    - a few days to release the system
  - **Selection list**

1. Rewrite query and application
2. Wait for 2 years: new PG version would improve planner.
3. Give up the migration project.
4. Use the forbidden fruit.

What is the “forbidden fruit”?

Innovative R&D by NTT

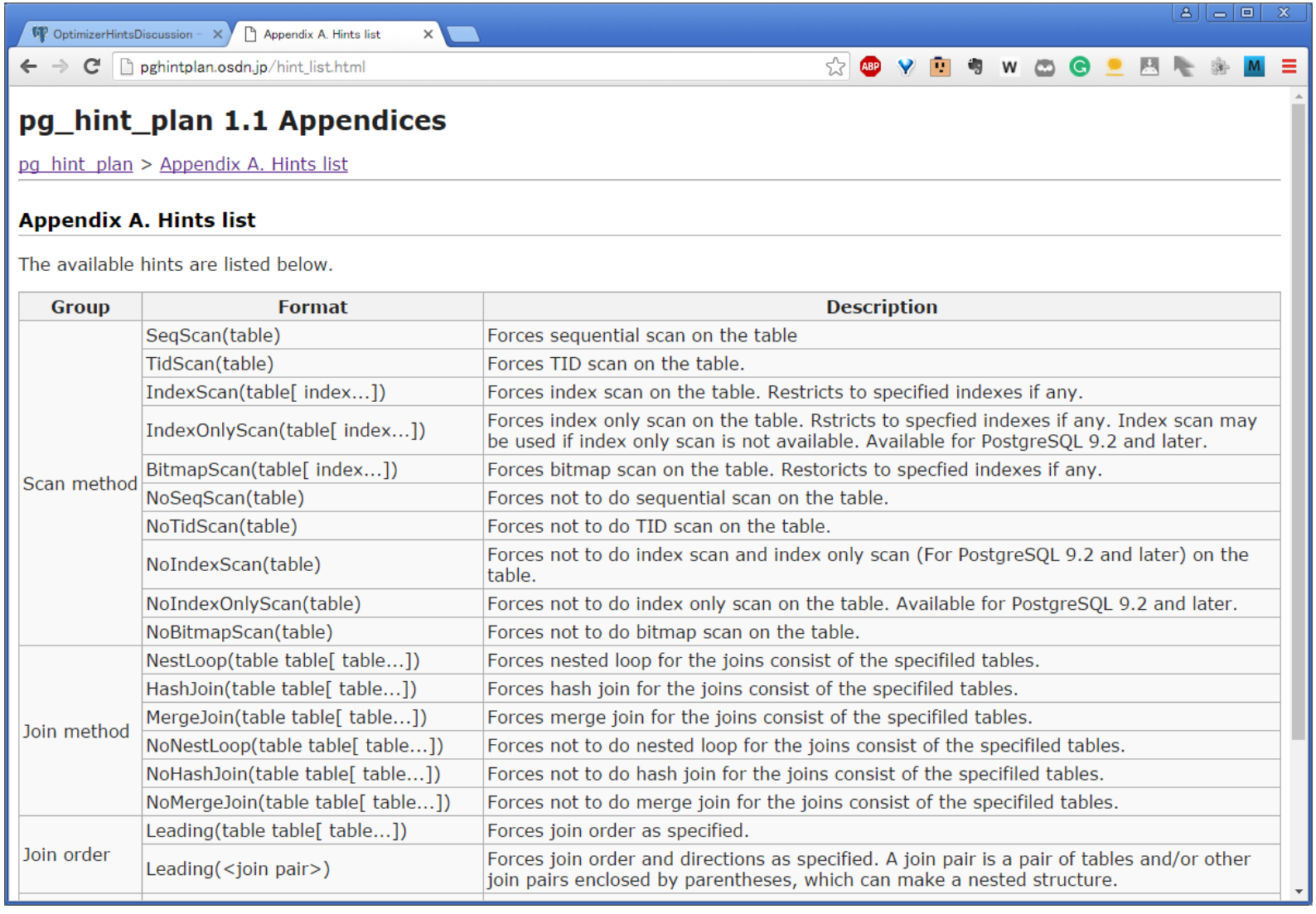
- **A special tool brewed in-house:**
  - This tool can control a plan using **optimizer hints**.
  - The name is **pg_hint_plan**.
    - tool for if the planner doesn't give you desired plans.
    - **able to control individual part of the plan!**
      set enable_* parameters can not do that.
  - Examples
    - /*+ IndexScan(foo) */
    - /*+ Leading((foo bar)) */
    - /*+ HashJoin(foo bar) */

NTT

pg_hint_plan provides a lot of Hints!

Innovative R&D by NTT

- **Around 20 kinds of hints.**



OptimizerHintsDiscussion | Appendix A. Hints list

pghintplan.osdn.jp/hint_list.html

pg_hint_plan 1.1 Appendices

pg_hint_plan > Appendix A. Hints list

Appendix A. Hints list

The available hints are listed below.

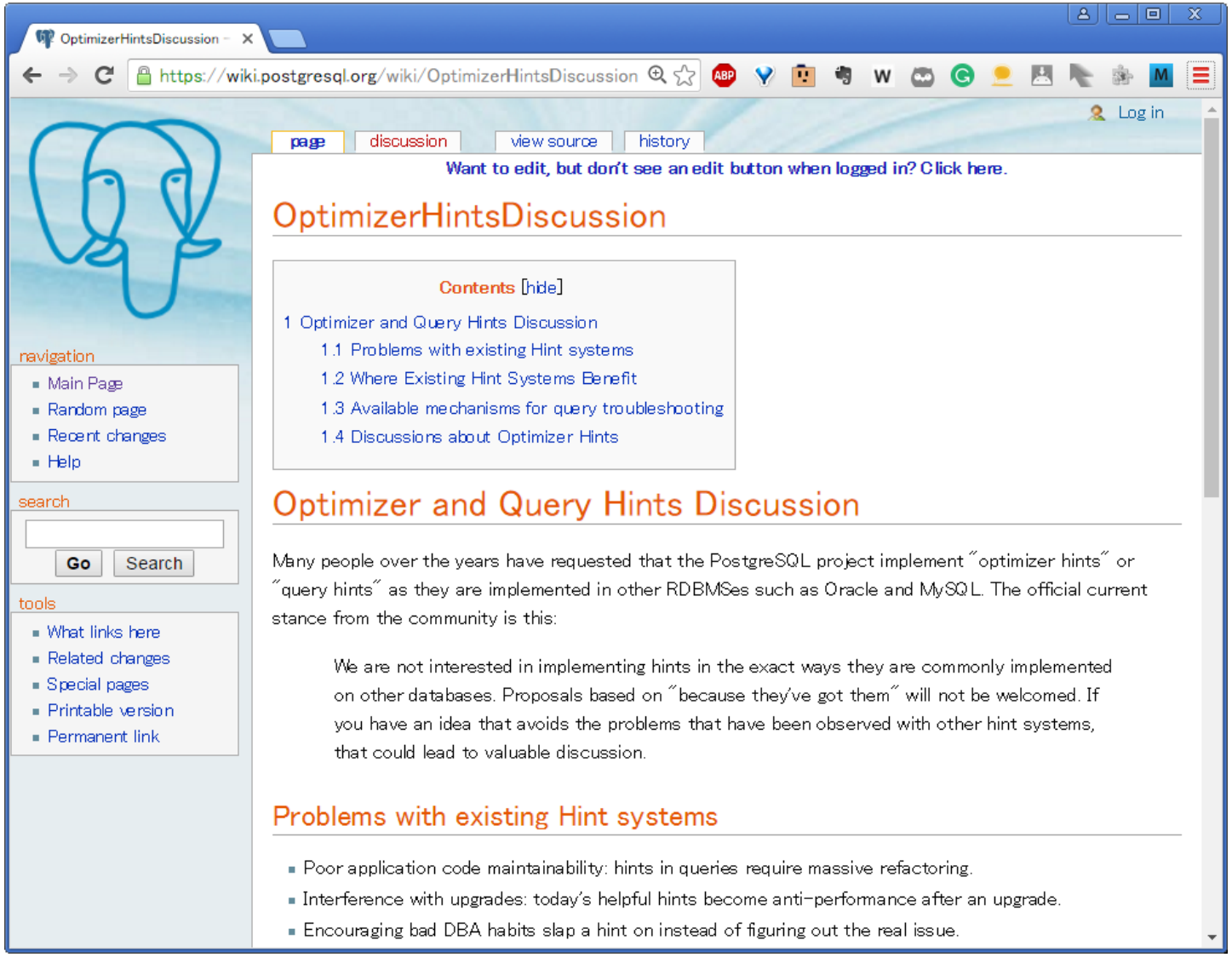
Group	Format	Description
Scan method	SeqScan(table)	Forces sequential scan on the table
	TidScan(table)	Forces TID scan on the table.
	IndexScan(table[index...])	Forces index scan on the table. Restricts to specified indexes if any.
	IndexOnlyScan(table[index...])	Forces index only scan on the table. Rstricts to specfied indexes if any. Index scan may be used if index only scan is not available. Available for PostgreSQL 9.2 and later.
	BitmapScan(table[index...])	Forces bitmap scan on the table. Restoricts to specfied indexes if any.
	NoSeqScan(table)	Forces not to do sequential scan on the table.
	NoTidScan(table)	Forces not to do TID scan on the table.
	NoIndexScan(table)	Forces not to do index scan and index only scan (For PostgreSQL 9.2 and later) on the table.
	NoIndexOnlyScan(table)	Forces not to do index only scan on the table. Available for PostgreSQL 9.2 and later.
	NoBitmapScan(table)	Forces not to do bitmap scan on the table.
Join method	NestLoop(table table[table...])	Forces nested loop for the joins consist of the specifiled tables.
	HashJoin(table table[table...])	Forces hash join for the joins consist of the specifiled tables.
	MergeJoin(table table[table...])	Forces merge join for the joins consist of the specifiled tables.
	NoNestLoop(table table[table...])	Forces not to do nested loop for the joins consist of the specifiled tables.
	NoHashJoin(table table[table...])	Forces not to do hash join for the joins consist of the specifiled tables.
	NoMergeJoin(table table[table...])	Forces not to do merge join for the joins consist of the specifiled tables.
Join order	Leading(table table[table...])	Forces join order as specified.
	Leading(<join pair>)	Forces join order and directions as specified. A join pair is a pair of tables and/or other join pairs enclosed by parentheses, which can make a nested structure.

NTT

Dev. Community doesn't like it…

Innovative R&D by NTT

- We know the advantage/disadvantage of it.



NTT

Decision to use hints

- **discussed the advantage/disadvantage and decided to use Hints on the project!**

Disadvantage

- Poor application code maintainability:
  - hints in queries require massive refactoring.

Doesn't matter

- Does not scale with data size:
  - the hint that's right when a table is small is like[illegible]wrong when it gets larger.

Doesn't matter

Advantage

- **Prevents optimizer failure:**
  - Implementation failure (known issues)
  - Theoretical failure (estimation limits, n^2 correlation problem)

It matters

※This is part of an excerpt of the list on the wiki.

We solved using the hints.

- **The inefficient plans were revised to the efficient plans and were stable.**
  - several days: bad -> several minutes: good
- **We were able to achieve stable batch performance.**
- **The system could accomplish batch performance stability until EOL.**

pg_hint_plan in our cases

- **Our problem is classified into the 2 categories.**
  - **Planner Row-Count Errors**
    1. **Can't See Through WITH**
    2. **Join selectivity doesn't know about cross-table correlations**

Thanks to Rovert Haas to share the planner error list.
https://sites.google.com/site/robertmhaas/query-performance/planner-row-count-errors

pg_hint_plan in our cases: Case1

1. Row count error by WITH

- **This test case is a example for row count error by WITH.**
- **The difference is the limit rows: 199 or 200.**
- **We can expect same plan and execution time.**

Limit 199

```
explain analyze with x as (select * from t1 limit 199)
select * from
    (select * from t1 where a in (select a from x)) tmp,t2
where tmp.a=t2.a;
```

Limit 200

```
explain analyze with x as (select * from t1 limit 200)
select * from
    (select * from t1 where a in (select a from x)) tmp,t2
where tmp.a=t2.a;
```

- These sql just to illustrate the problem.

Row count error by WITH (cont.)

- **Results of Explain analyze**
  - **The plan and execution time are different.**

Limit 199

```
Nested Loop  (cost=9.76..17157.31 rows=199 wi   =16)
                  (actual time=0.234..195.086 rows=199 l  ops=1)
(snip)
Execution time: 195.155 ms
```

error?

Limit 200

```
Hash Join  (cost=30839.81..48153.89 rows=500000 width=16)
                  (actual time=249.394..303.259 rows=200 loops=1)
(snip)
Execution time: 303.335 ms
```

x 1.5?!

- **By the Row count error,**
  the plan and time are changed. Why?

Row count error by WITH (cont.)

- **Cause?**
  - The CTE **doesn't have statistics** since it is a temporary table.
  - The Rows estimation is calculated using default values.
  - The Planner **can't optimize the plan**.
- **Solution**
  - pg_hint_plan can control the plan manually.
    - /*+ Leading((t1 x)) Hashjoin(t1 x) */

Row count error by WITH (cont.)

- **The result of query using the Hint.**

before

Limit 200

```
-----------------------------------------------------------------------------------------------------
Hash Join  (cost=30839.81..48153.89 rows=500000 width=16)
                (actual time=249.394..303.259 rows=200 loops=1)
(snip)
Execution time: 303.335 ms
```

Same rows

after

Limit 200 + Hints

```
-----------------------------------------------------------------------------------------------------
 Nested Loop  (cost=10000000012.31..10000258778.8
                                          rows=500000 width=16)
                (actual time=0.333..197.523 rows=200 loops=1)
(snip)
Execution time: 197.628 ms
```

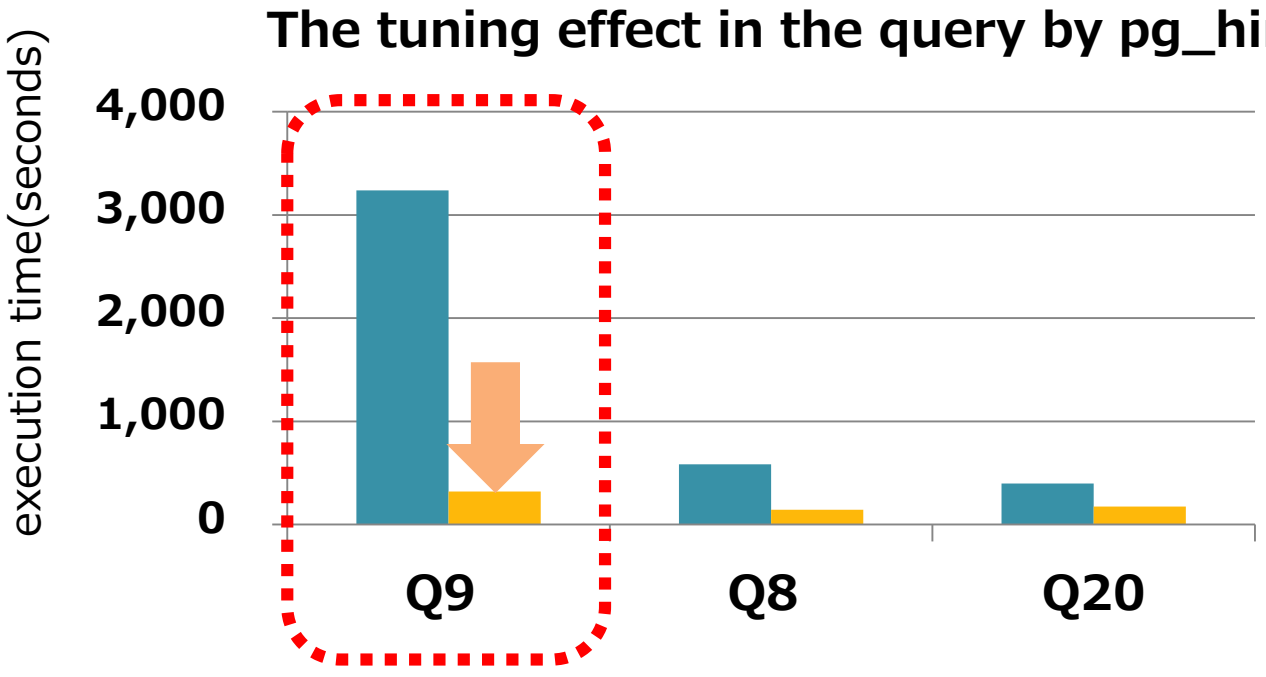
By changing the plan, the Time decreased.

※ Limit 199: 195.155ms

pg_hint_plan in our cases: Case2

2. Row count error by cross-table correlations

- TPC-h's Q9 is a famous example.
- We can reduce the execution time using the hints.



- **Original**
- **Tuning by hints**
  - DBT-3
  - scale factor=24
  - HP DL380 G7 32GB
  - PostgreSQL 9.4.1
  - Shared buffers=3GB

Row count error by cross-table correlations (cont.)

Innovative R&D by NTT

- **Cause?**
  - The planner can't use Join selectivity with cross-table correlations.
  - It excessively expected a small number of joined rows.
  - In this situation, a Nested Loop is not efficient.
    - Ex. Estimated rows: 1, Actual rows: 10000000
- **Solution**
  - **pg_hint_plan can provide the efficient plan.**
  - Cf. this problem will be fixed soon.
    Keep going! Tomas, Horiguchi and hackers! ☺

    http://www.postgresql.org/message-id/543AFA15.4080608@fuzzy.cz

NTT

Challenges

1. To run OLTP and OLAP without interfering each other 
2. To guarantee performance on cloud 
3. **To ensure batch performance stability until EOL** 

Summary: challenges and solutions

Innovative R&D by NTT

1. To run OLTP and OLAP without interfering each other

- Creating the read-replica using SR easily.
  Make sure 2 parameters for OLAP.

2. To guarantee performance on the cloud

- Verifying I/O performance is the key to success.
  PostgreSQL can use on the cloud infrastructure.

3. To ensure batch performance stability until EOL

- In the case of the Optimizer failure,
  must use pg_hint_plan for controlling a plan.
  It can become a strong weapon for users.

NTT

Project Goals achieved

- **The migration challenges were successful.**
- **The system keeps working stably since May 2015.**

4. Thoughts about the future of PostgreSQL

Thoughts about the future of PostgreSQL

- **Hints vs. Planner improvements**
  - The planner's estimation is sometimes wrong since limitation/specification.
    - By reducing the mistake, many users can reduce the system development costs.
  - PostgreSQL has "set enable_*" parameters.
    but it's not useful to revise an inefficient plan.
    Because application scope is too broad – affects all nodes of a given type in the plan..
  - pg_hint_plan is able to control individual part of the plan.
  - **Should we implement "the Optimizer Hints" to PostgreSQL core?**

Thoughts (cont.)

Innovative R&D by NTT

OR

NTT

Requests Planner improvements

- **I have 2 ideas for the planner**

1. Feedback loop for planning

We can get a new efficient plan using a past plan result:
"Actual rows" , "Actual time" and so on.
It is similar to PDCA cycle.

2. Plan cache/Plan table

Choosing and freezing an efficient plan from a plan cache/table is useful for stably performance.
This feature provide a plan management to user.

1. Feedback loop for planning

Innovative R&D by NTT

- **The use cases of feedback loop**

Examples

1.1. Creating an alternative plan

if the plan has huge differences between
the estimation records and the actual records.
The planner should create an alternative plan.

Nested loop <-> Hash join

1.2. Validating the Statistics for planning

The planner doesn't know the statistics are accurate.
If the statistics have Correctness factor/Risk factor
which is results of validating.
The planner can use these factors to consider
alternative plans.

NTT

2. Plan cache/Plan table

- **The use cases of Plan cache/table**

Examples

2.1. Monitoring a plan using Plan history

We looked at log to check when plan changes.
The plan cache/table can investigate it easily.

2.2. Choose plan from Plan history

You can choose a plan which you desired manually, then the planner will use the plan always.
In addition, We can possible to get suggestion of efficient plan using analyzed the historical data automatically.

5. Conclusion

5. Conclusion

- In this talk, I have shared my experience on the migration project in NTT and my thoughts about the future of PostgreSQL.

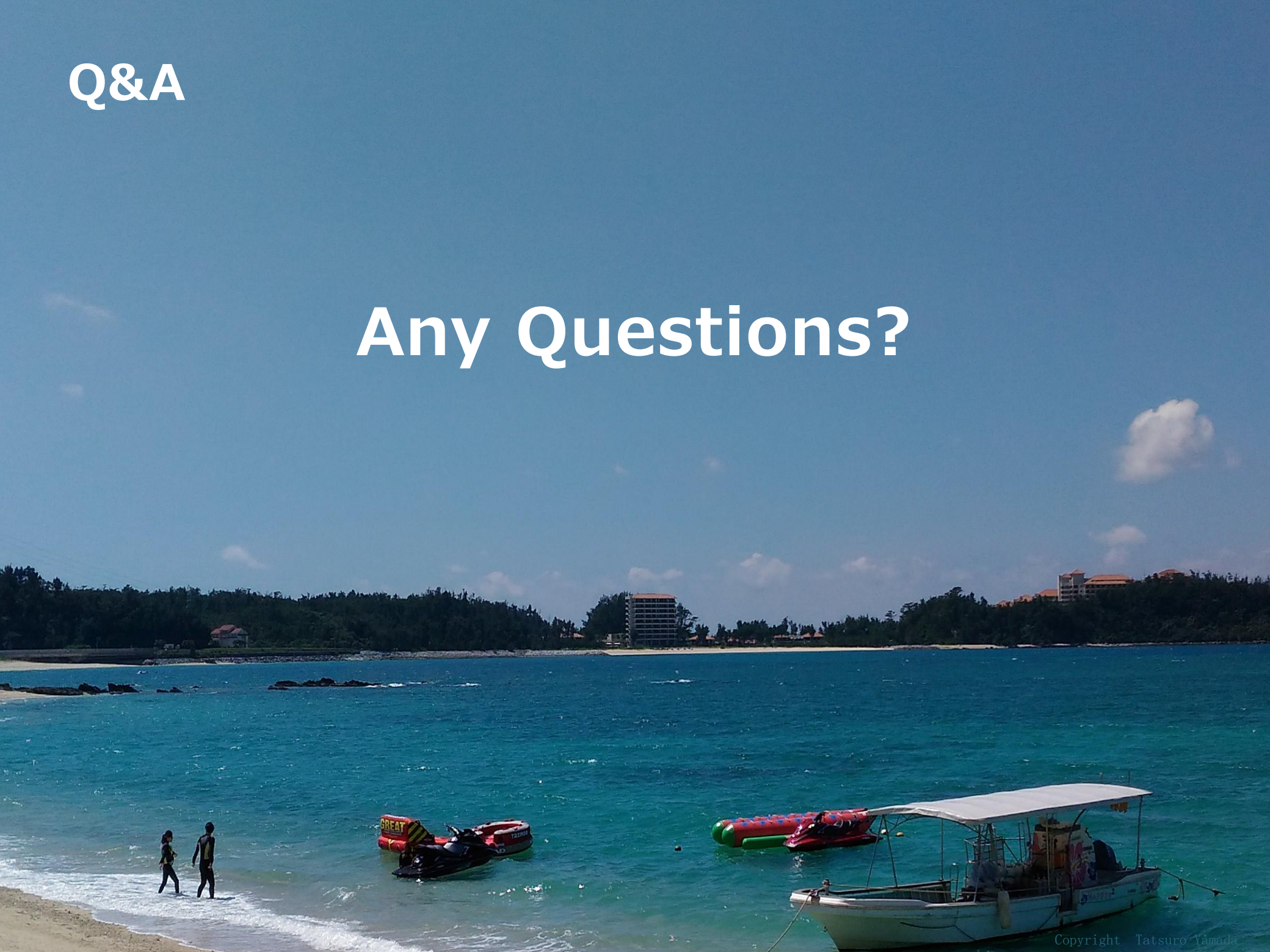
- I hope that I was able to prove that PostgreSQL can be used in your mission-critical fields.

Thank you

Plan to succeed

NTT OSS Center

Q&A



References

- **NTT OSS Center on GitHub**
  - https://github.com/ossc-db
    - pg_reorg
    - pg_rman
    - pg_bulkload
    - pg_hint_plan
    - pg_dbms_stats
    - pg_store_plans
    - dblink_plus
    - db_syntax_diff
    - syncdb
- **SourceForge**
  - https://sourceforge.net/projects/pgstatsinfo/
    - pg_statsinfo
    - pg_stats_reporter

Appendix

- Optimizer Hints Discussion
- Row count error by WITH

Optimizer Hints Discussion

Innovative R&D by NTT

- https://wiki.postgresql.org/wiki/OptimizerHintsDiscussion

- Demerits

No.	Subject	Detail
1	**Poor application code maintainability**	hints in queries require massive refactoring.
2	**Interference with upgrades**	today's helpful hints become anti-performance after an upgrade.
3	**Encouraging bad DBA habits slap a hint on instead of figuring out the real issue.**	-
4	**Does not scale with data size**	the hint that's right when a table is small is likely to be wrong when it gets larger.
5	**Failure to actually improve query performance**	most of the time, the optimizer is actually right.
6	**Interfering with improving the query planner**	people who use hints seldom report the query problem to the project.

NTT

Optimizer Hints Discussion

Innovative R&D by NTT

- Merits

No.	Subject	Detail
1	**"One-shot" issues, such as annual or one-time reports, for which maintainability is not a concern**	-
2	**Ability to "test" various execution paths in detail and see how the optimizer is working (or not)**	-
3	**Optimizer failure**	Implementation failure (known issues)
		Theoretical failure (estimation limits, n^2 correlation problem)

NTT

Optimizer Hints Discussion

- **discussed the Fit/Gap on the list.**
  - **Demerits of optimizer hint**

No.	Demerits	FIT/GAP and Reasons
1	Poor application code maintainability	GAP
		Only using a few hints.
2	Interference with upgrades	GAP
		Do not upgrade if there is no fatal bug.
3	Encouraging bad DBA habits slap a hint on instead of figuring out the real issue.	GAP
		We used the hint on some queries which could not be improved.
4	Does not scale with data size	GAP
		Confirmed by performance tests using the data amount of EOL time of the system.
5	Failure to actually improve query performance	GAP
		Important for performance to be stable while meeting the performance requirements than the peak performance
6	Interfering with improving the query planner	GAP
		No problem because it is shared by PGCon or -hackers.

Optimizer Hints Discussion

- **The hint gives the merit to the project.**
  - **Merits of optimizer hint**

No.	Merits	FIT/GAP and Reasons
1	"One-shot" issues, such as annual or one-time reports, for which maintainability is not a concern	GAP
		-
2	Ability to "test" various execution paths in detail and see how the optimizer is working (or not)	FIT
		We use Hints for tuning the plan
3	Prevents optimizer failure	FIT
		We'd like to prevent optimizer failure.

- **Decided to use the hint.**

Row count error by WITH

```
create table t1 (a int, b int);

insert into t1 (select a, random() * 1000 from
generate_series(0, 999999) a);

create index i_t1_a on t1 (a);
analyze t1;

create table t2 (a int, b int);

insert into t2 (select a, random() * 1000 from
generate_series(0, 999999) a);

create index i_t2_a on t2 (a);
analyze t2;
```

Row count error by WITH

- **explain analyze with x as (select * from t1 limit 199) select * from (select * from t1 where a in (select a from x)) tmp,t2 where tmp.a=t2.a;**

```
                                      QUERY PLAN
-----------------------------------------------------------------------------------------
-
 Nested Loop  (cost=9.76..17157.31 rows=199 width=16) (actual time=0.234..195.086 rows=199 loops=1)
   CTE x
     ->  Limit  (cost=0.00..2.87 rows=199 width=8) (actual time=0.008..0.079 rows=199 loops=1)
           ->  Seq Scan on t1 t1_1  (cost=0.00..14425.00 rows=1000000 width=8)
                                     (actual time=0.007..0.044 rows=199 loops=1)
   ->  Hash Semi Join  (cost=6.47..17058.68 rows=199 width=12)
                       (actual time=0.224..194.404 rows=199 loops=1)
         Hash Cond: (t1.a = x.a)
         ->  Seq Scan on t1  (cost=0.00..14425.00 rows=1000000 width=8)
                             (actual time=0.014..92.474 rows=1000000 loops=1)
         ->  Hash  (cost=3.98..3.98 rows=199 width=4) (actual time=0.201..0.201 rows=199 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 15kB
               ->  CTE Scan on x  (cost=0.00..3.98 rows=199 width=4)
                                  (actual time=0.010..0.158 rows=199 loops=1)
   ->  Index Scan using i_t2_a on t2  (cost=0.42..0.47 rows=1 width=8)
                                      (actual time=0.003..0.003 rows=1 loops=199)
         Index Cond: (a = t1.a)
 Planning time: 0.401 ms
 Execution time: 195.155 ms
(14 rows)
```

Row count error by WITH

- explain analyze with x as (select * from t1 limit 200) select * from (select * from t1 where a in (select a from x)) tmp,t2 where tmp.a=t2.a;

```
                                          QUERY PLAN
--------------------------------------------------------------------------------------------------------
-
 Hash Join  (cost=30839.81..48153.89 rows=500000 width=16) (actual time=249.394..303.259 rows=200
loops=1)
   Hash Cond: (x.a = t1.a)
   CTE x
     ->  Limit  (cost=0.00..2.88 rows=200 width=8) (actual time=0.015..0.064 rows=200 loops=1)
           ->  Seq Scan on t1 t1_1  (cost=0.00..14425.00 rows=1000000 width=8)
                                    (actual time=0.013..0.037 rows=200 loops=1)
   ->  Nested Loop  (cost=4.92..1653.00 rows=500000 width=12)
                    (actual time=0.230..0.584 rows=200 loops=1)
         ->  HashAggregate  (cost=4.50..6.50 rows=200 width=4)
                            (actual time=0.217..0.255 rows=200 loops=1)
               Group Key: x.a
               ->  CTE Scan on x  (cost=0.00..4.00 rows=200 width=4)
                                  (actual time=0.017..0.124 rows=200 loops=1)
         ->  Index Scan using i_t2_a on t2  (cost=0.42..8.22 rows=1 width=8)
                                            (actual time=0.001..0.001 rows=1 loops=200)
               Index Cond: (a = x.a)
   ->  Hash  (cost=14425.00..14425.00 rows=1000000 width=8)
             (actual time=248.859..248.859 rows=1000000 loops=1)
         Buckets: 131072  Batches: 16  Memory Usage: 3471kB
         ->  Seq Scan on t1  (cost=0.00..14425.00 rows=1000000 width=8)
                             (actual time=0.009..92.095 rows=1000000 loops=1)
 Planning time: 0.611 ms
 Execution time: 303.335 ms
(16 rows)
```

Row count error by WITH

- explain analyze /*+ Leading((t1 x)) Hashjoin(t1 x) */ with x as (select * from t1 limit 200) select * from (select * from t1 where a in (select a from x)) tmp,t2 where tmp.a=t2.a;

```
                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
-
 Nested Loop  (cost=10000000012.31..10000258778.89 rows=500000 width=16) (actual
time=0.333..197.523 rows=200 loops=1)
   CTE x
     ->  Limit  (cost=0.00..2.88 rows=200 width=8) (actual time=0.008..0.062 rows=200 loops=1)
           ->  Seq Scan on t1 t1_1  (cost=0.00..14425.00 rows=1000000 width=8)
                                    (actual time=0.007..0.029 rows=200 loops=1)
   ->  Hash Join  (cost=9.00..18186.00 rows=500000 width=12)
                  (actual time=0.315..196.816 rows=200 loops=1)
         Hash Cond: (t1.a = x.a)
         ->  Seq Scan on t1  (cost=0.00..14425.00 rows=1000000 width=8)
                              (actual time=0.016..93.165 rows=1000000 loops=1)
         ->  Hash  (cost=6.50..6.50 rows=200 width=4) (actual time=0.290..0.290 rows=200 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 16kB
               ->  HashAggregate  (cost=4.50..6.50 rows=200 width=4)
                                  (actual time=0.206..0.248 rows=200 loops=1)
                     Group Key: x.a
                     ->  CTE Scan on x  (cost=0.00..4.00 rows=200 width=4)
                                        (actual time=0.012..0.139 rows=200 loops=1)
   ->  Index Scan using i_t2_a on t2  (cost=0.42..0.47 rows=1 width=8)
                                      (actual time=0.003..0.003 rows=1 loops=200)
         Index Cond: (a = t1.a)
 Planning time: 0.398 ms
 Execution time: 197.628 ms
(16 rows)
```

pg_hint_plan example

Nested Loop => Hash Join

```
# EXPLAIN SELECT
#    FROM pgbench_branches b
#    JOIN pgbench_accounts a ON b.bid = a.bid
#   ORDER BY a.aid;
---------------------------
 Nested Loop (c
   Join Filter: (b.bid =
   -> Index Scan
   -> Materialize
        -> Seq Scan c
```

```
# /*+
#     HashJoin(a b)
#     SeqScan(a)
#  */
# EXPLAIN SELECT
#    FROM pgbench_branches b
#    JOIN pgbench_accounts a ON b.bid = a.bid
#   ORDER BY a.aid;
--------------------------------------------------------------------------------
 Sort  (cost=12320.84..12570.84 rows=100000 width=4)
   Sort Key: a.aid
   -> Hash Join  (cost=1.02..4016.02 rows=100000 width=4)
        Hash Cond: (a.bid = b.bid)
        -> Seq Scan on pgbench_accounts a  (cost=0.00..2640.00 rows=100000 width=8)
        -> Hash  (cost=1.01..1.01 rows=1 width=4)
             -> Seq Scan on pgbench_branches b  (cost=0.00..1.01 rows=1 width=4)
```

これがヒント句。
ここではハッシュ結合と
シーケンシャルスキャンを指定。