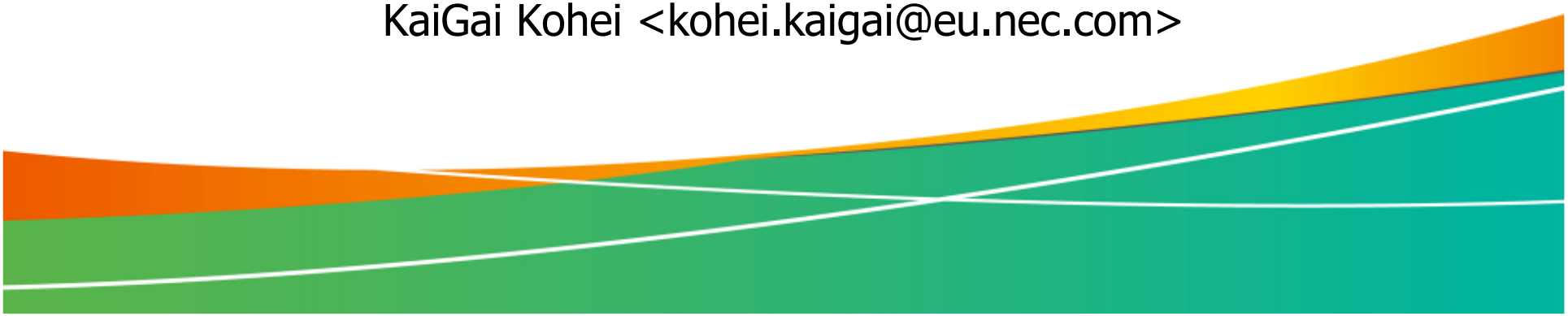
Empowered by Innovation NEC

PostgreSQL Developer Meeting 2011
~Security Features towards v9.2~

NEC Europe Ltd,
SAP Global Competence Center

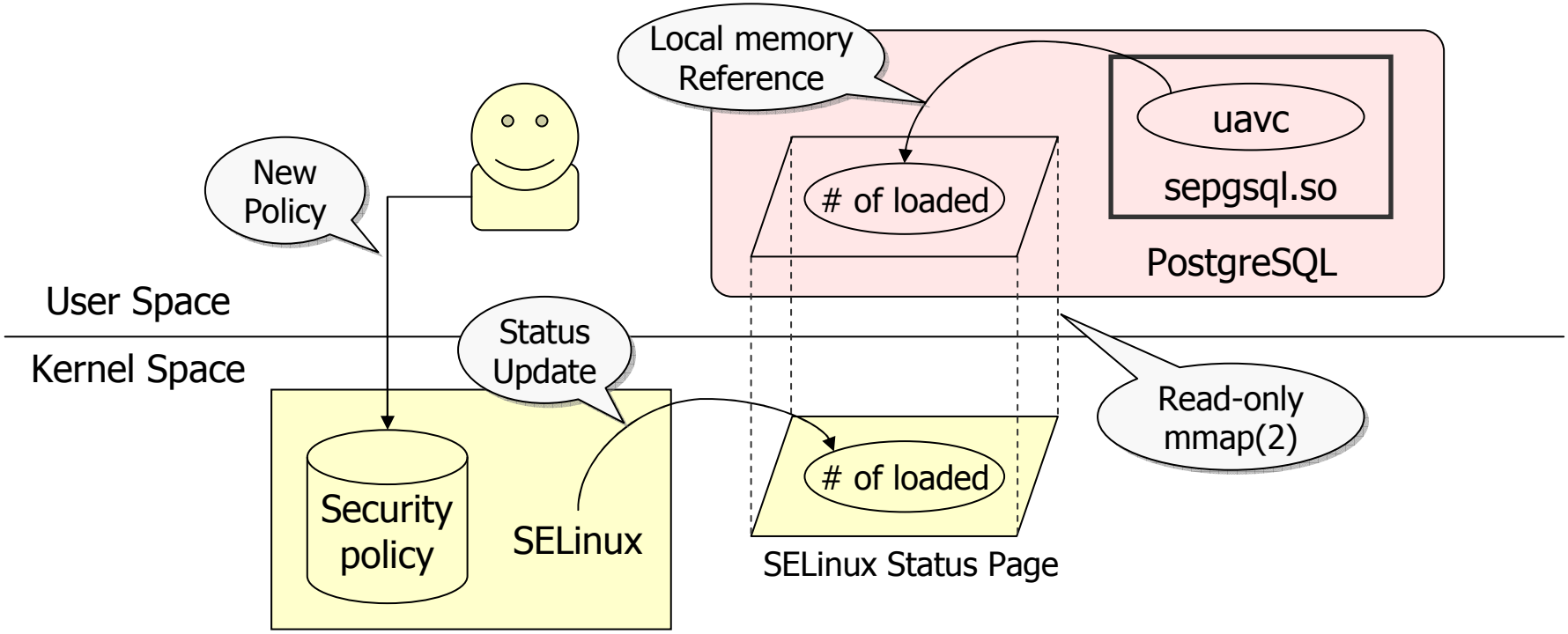
KaiGai Kohei <kohei.kaigai@eu.nec.com>



Linux kernel updates for SE-PostgreSQL (1/2)

SELinux kernel status page

- Background: System call invocations were needed for each access control to validate cached previous decisions in userspace.
  ➔ However, it took unignorable cost for performance.
- Kernel allowed applications mmap(read-only) the status page of SELinux.

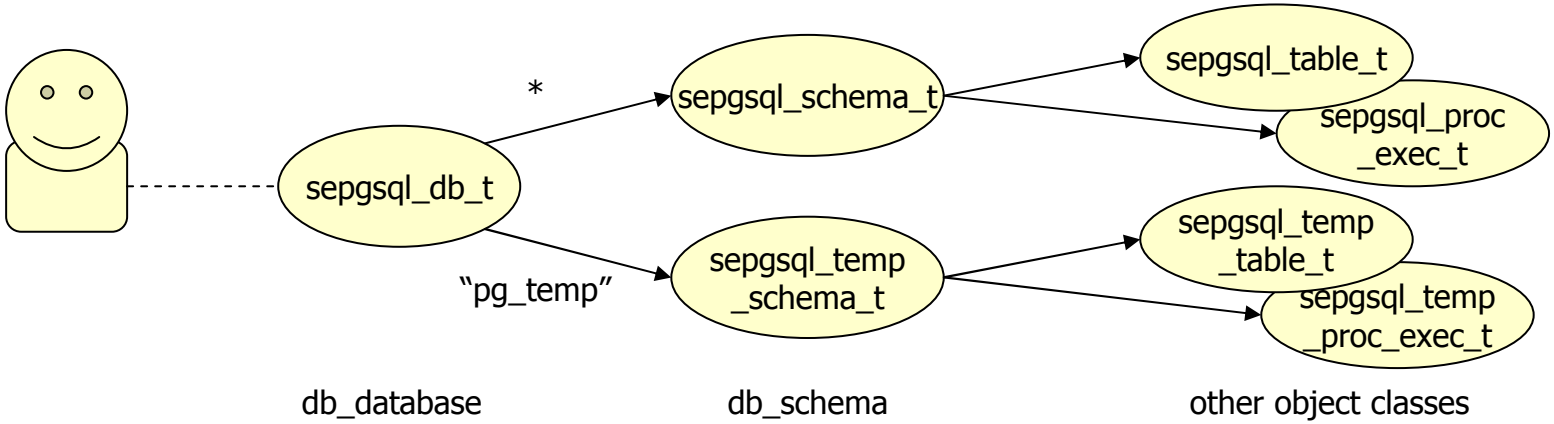


Linux kernel updates for SE-PostgreSQL (2/2)

Named TYPE_TRANSITION Rule

- Background: Default security label of a new object was determined based on a pair of security labels of client and upper object.
  ➔ However, inconvenient in some cases
- Policy allowed to define exceptional default label when a new object has particular name. (E.g "pg_temp" as a schema object)

```
TYPE_TRANSITION user_t sepgsql_db_t :
        db_schema sepgsql_schema_t;
TYPE_TRANSITION user_t sepgsql_db_t :
        db_schema sepgsql_temp_schema_t pg_temp;
```



Leaky VIEWs and RLS (1/3) – Problems

- According to the discussion a half years ago...

User-defined functions with side-effects and little-cost

- Unexpected order to evaluate qualifiers on a certain scan plan

  If a function to filter out scanned tuples has very small cost, the optimizer reorders them to evaluate. If it is smaller than view-definition, contents of tuples can be leaked via the functions.

  - It is reordered at `order_qual_clauses()`.

- Unexpected qualifier distributions into inside of join loop

  If a function to be performs outside of a join-loop references only one-side of them, the optimizer distributes the function inside of the join-loop. Then, it can leak the given argument prior to filtering out.

  - It is distributed at `distribute_qual_to_rels()`.

Iteration of proving PK/FK and UNIQUE constraint

Statistical based on invisible tuples

➔ Do we need to consider them as problems?

Leaky VIEWs and RLS (2/3) – Principles

Scope of the target

- The point is to clarify what is the point to be tackled.
- Row-level security shall prevent unprivileged users to reference the contents of invisible tuples directly.

Unreasonable scenario

- A user-supplied function can be executed earlier than other functions that filter-out invisible tuples, because it allows us to leak the given arguments come from invisible tuples.

Reasonable scenario

- Server stuff internally reference whole of the contents, as long as invisible tuples being kept hidden from users.
  - ✓ E.g) Hint for query optimization, PK/FK checks, Index access methods, ...
- User may infer contents of the invisible tuples, as long as it is forbidden to reference them directly
  - ✓ E.g) Iteration of PK/FK proving, Reference to statistical, ...
  - ✓ We also called it as "covert-channel"

Leaky VIEWs and RLS (3/3) – Solutions

Add `CREATE [SECURITY] VIEW` statement

- It gives a particular view special protection. If a certain subquery was expanded from a security view, the optimizer does not push down qualifiers into the subquery, even if it references only one-side of join-loop, except for obviously secure functions (E.g implementation of operators without error messages).
- A derivative idea is a new flag to mark whether a particular function is leakable, or not.
  - ➔ One headache is who can mark it on the functions in catalog/pg_proc.h ☹

Track nest-level of qualifiers to filter-out scanned tuples

- It gives the nest-level higher priority than estimated cost when optimizer reorders qualifiers. It ensures deeper level condition shall be checked earlier.

Our challenges in v9.2

- Security labels for shared database objects (database, role, ...)
  - Add pg_shseclabel, and extend SECURITY LABEL statement
- Userspace access vector cache
  - Add caching mechanism of access control decision
- DDL Permissions
  - Extend object_access_hook to take arguments
  - Add hooks around DDL checks (probably, step by step)
- Row-level access control
  - Fix leaky VIEWs problem
  - Support security labels (also ACLs) for user-defined tables

Empowered by Innovation