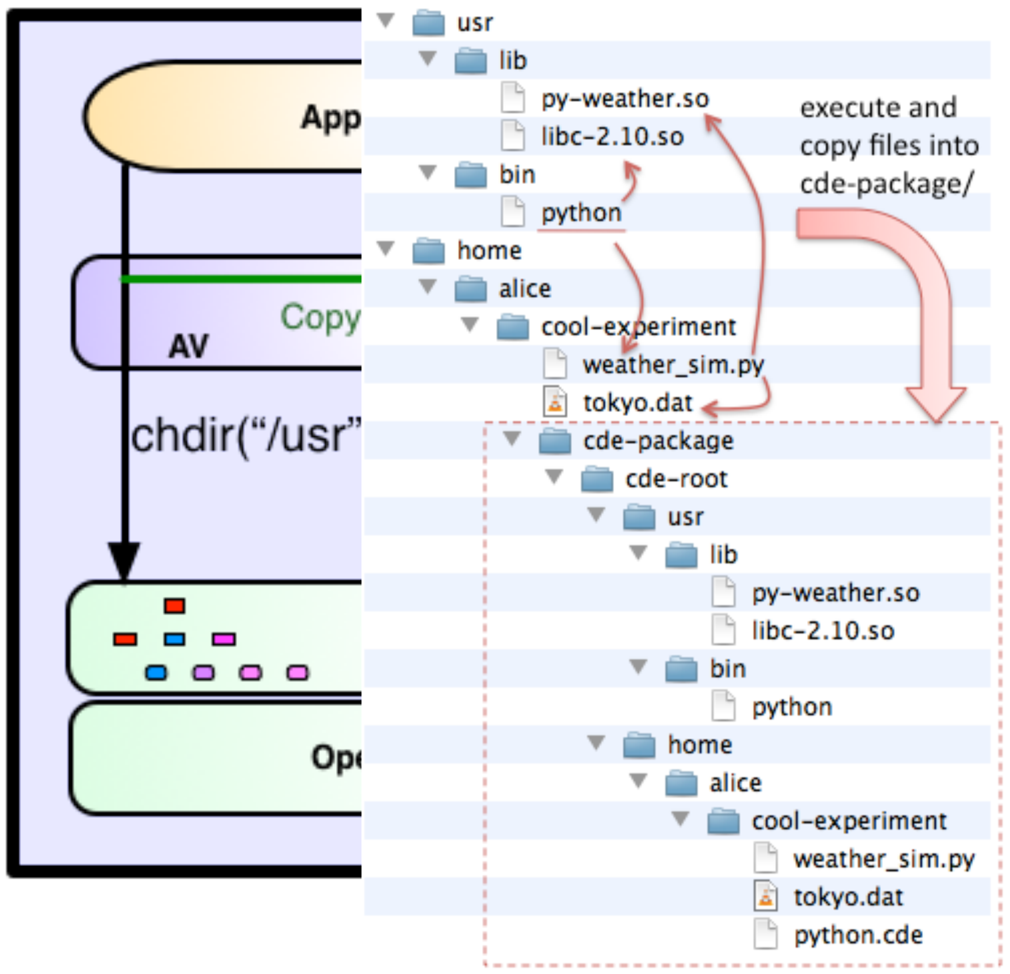
LDV: Light-weight Database Virtualization

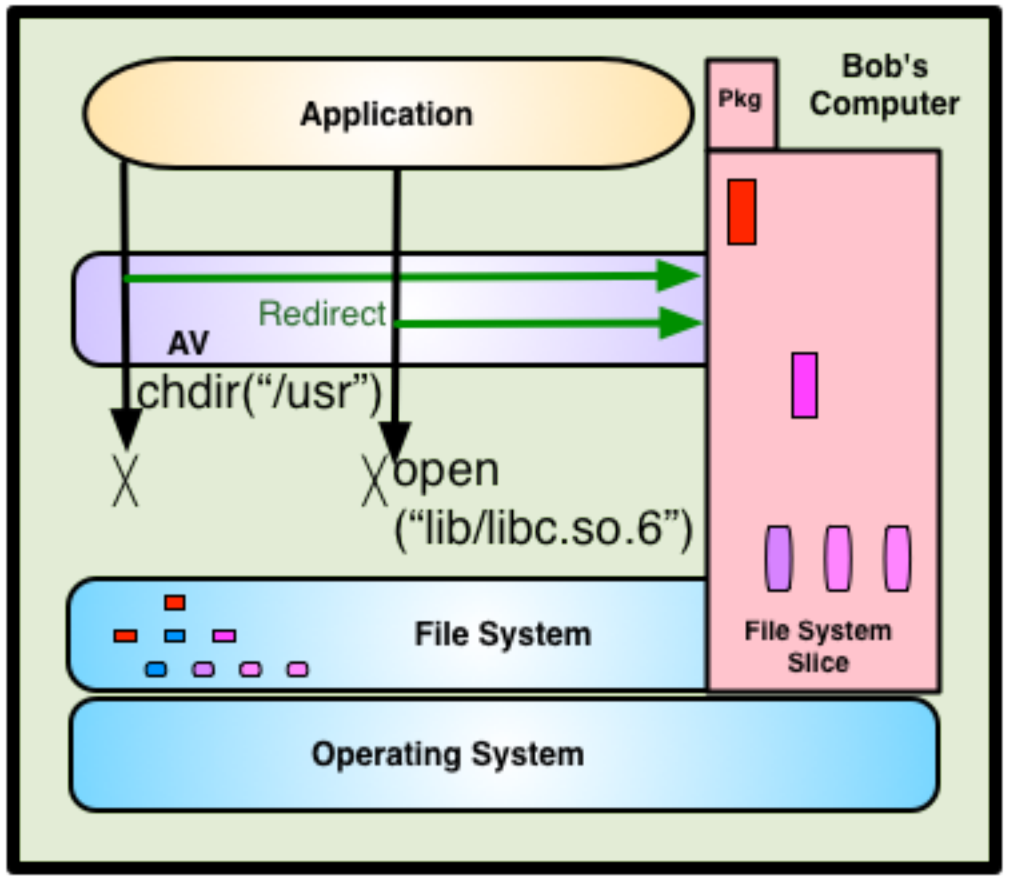
Quan Pham[2], Tanu Malik[1], Boris Glavic[3] and Ian Foster[1,2]
Computation Institute[1] and Department of Computer Science[2,3]
University of Chicago[1,2], Argonne National Laboratory[1]
Illinois Institute of Technology[3]

Application Virtualization

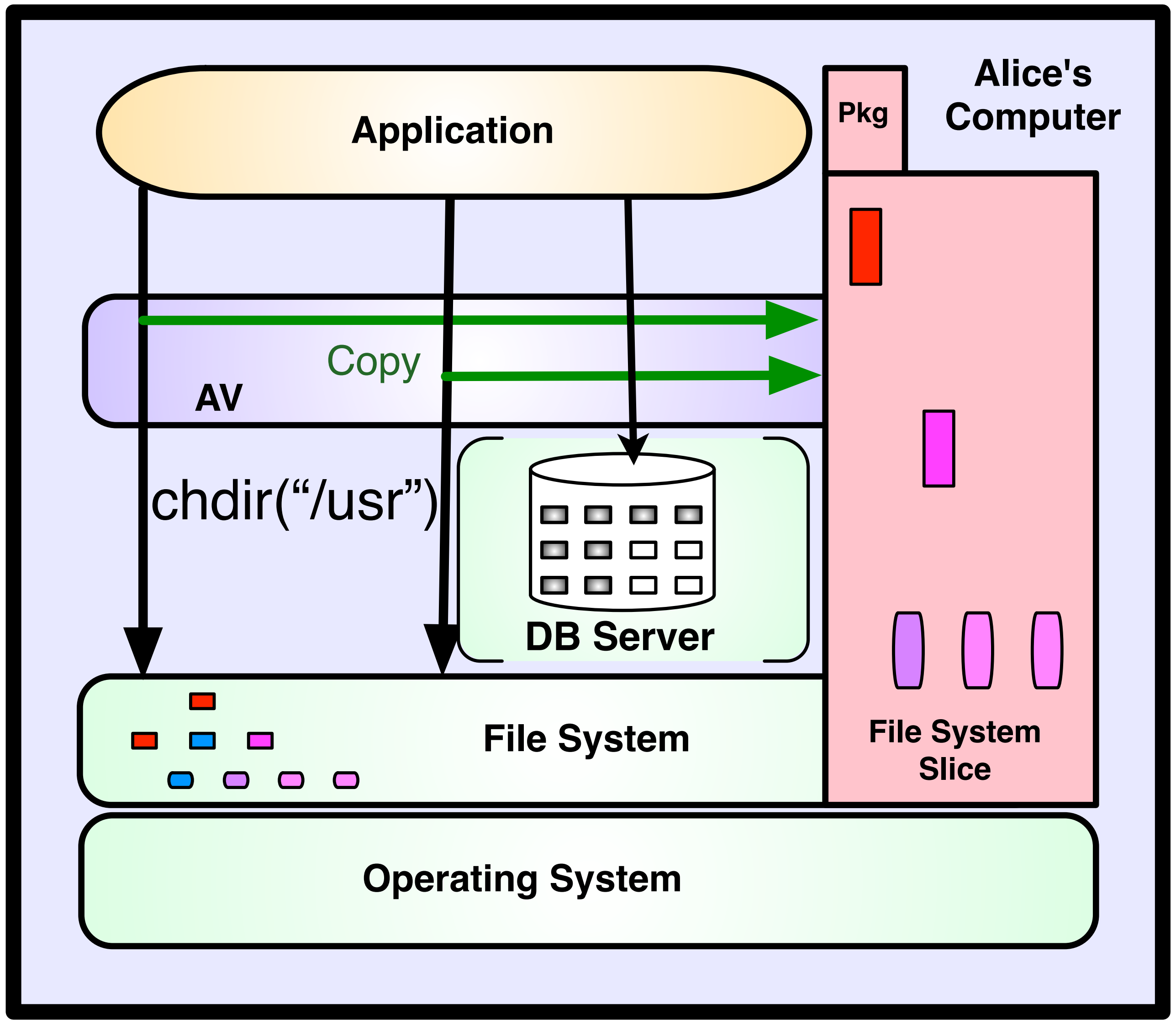
Alice's Machine



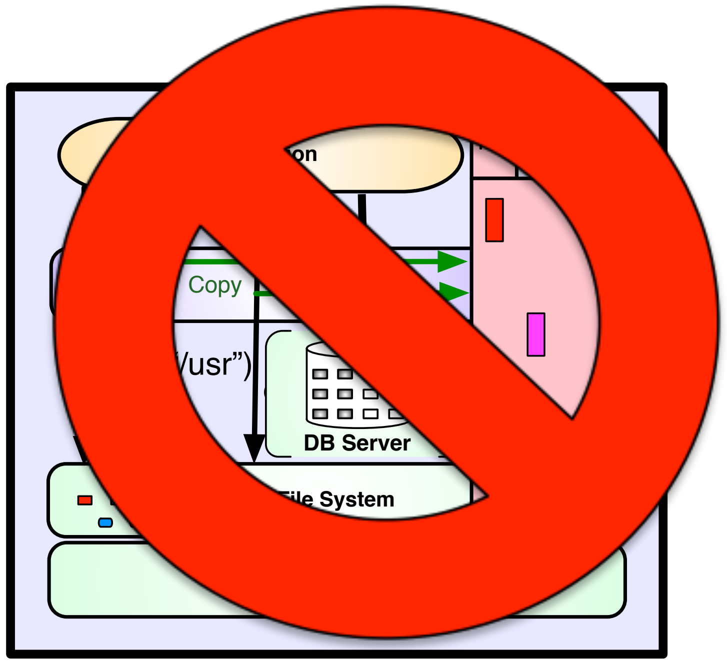
Bob's Machine



Application Virtualization for DB Applications



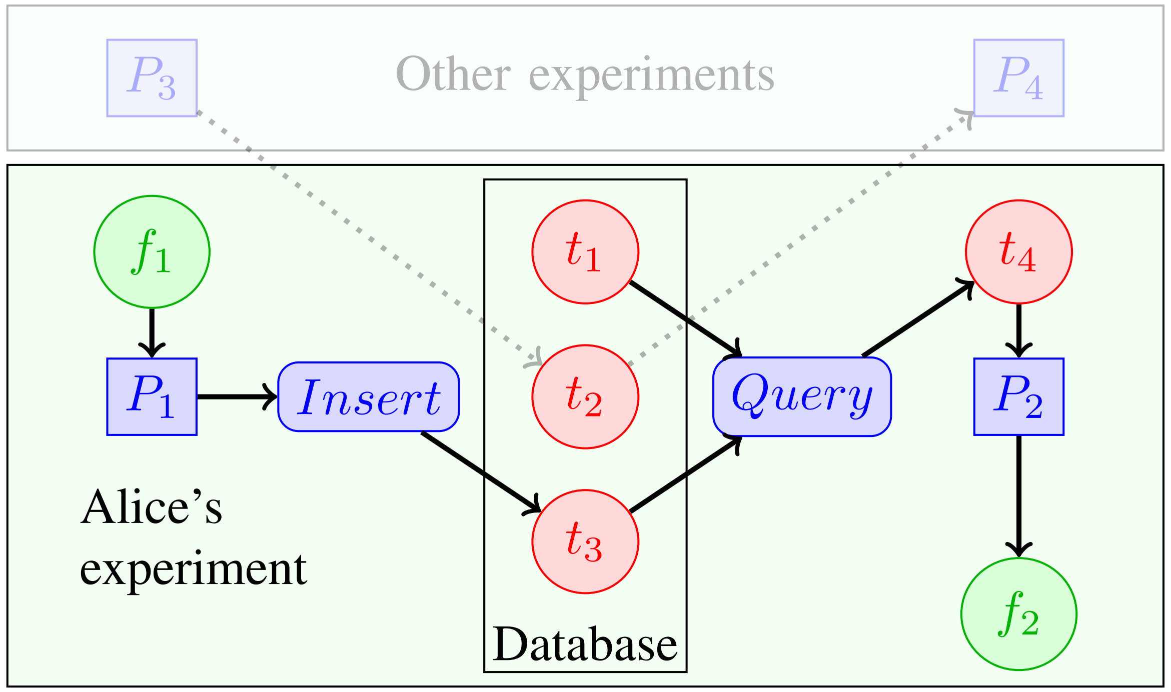
Application Virtualization for DB Applications



- Applications that interact with a relational database
- Examples:
  - Text-mining applications that download data, preprocess and insert into a personal DB
  - Analysis scripts using parts of a hosted database

Why doesn’t it work?

- Application virtualization methods are oblivious to semantics of data in a database system
- The database state at the time of sharing the application may not be the same as the start of the application



LDV: Light-weight Database Virtualization

- Goal: Easily and efficiently share and repeat DB applications.

Key Ideas

- DB application = Application (OS) part + DB part
  - Use data provenance to capture interactions from/to the application side to the database side
  - *Limited formal mechanisms so far to combine the two kinds of provenance models*
- Create a virtualized package that can be re-executed
  - Either include the server and data, or replay interactions (for licensed databases)
  - *No virtualization mechanism for database replay*

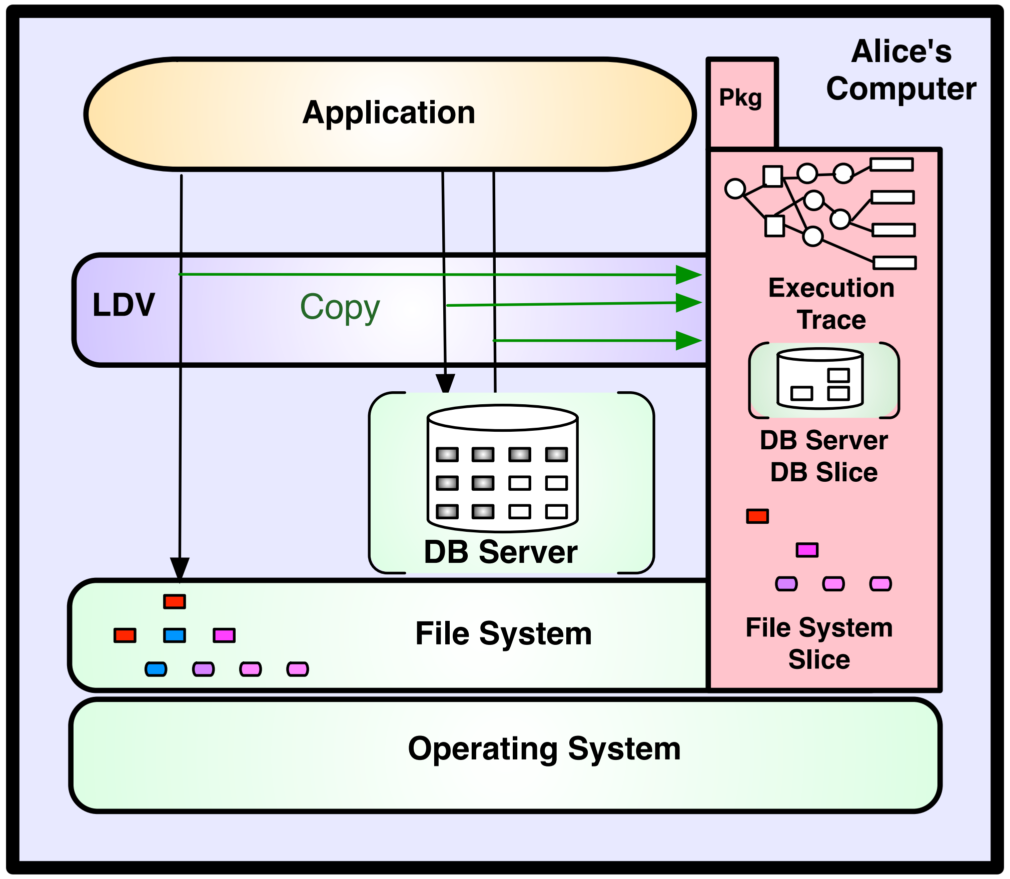
Related Work

- Application virtualization
  - Linux Containers, CDE[Usenix'11]
- Packaging with annotations
  - Docker
- Packaging with provenance
  - PTU[1][TaPP'13], ReproZip[TaPP'13], Research Objects
- Unified provenance models
  - based on program instrumentation [TaPP'12]

1 Q. Pham, T. Malik, and I. Foster. Using provenance for repeatability. In Theory and Practice of Provenance (TaPP), 2013.

How does LDV work?

Alice's Machine

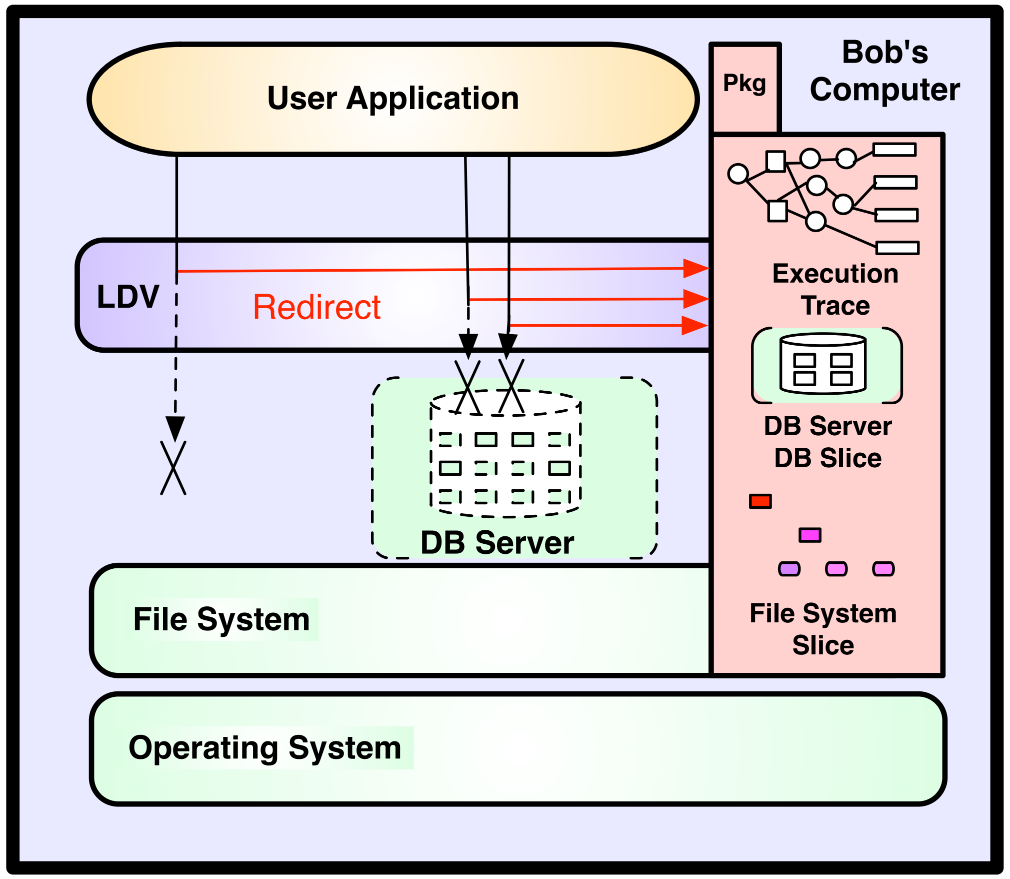


ldv-audit *db-app*

- Monitoring system calls
- Monitoring SQL
  - Server-included packages
  - Server-excluded packages
- Execution traces
- Relevant DB and filesystem slices

How does LDV work?

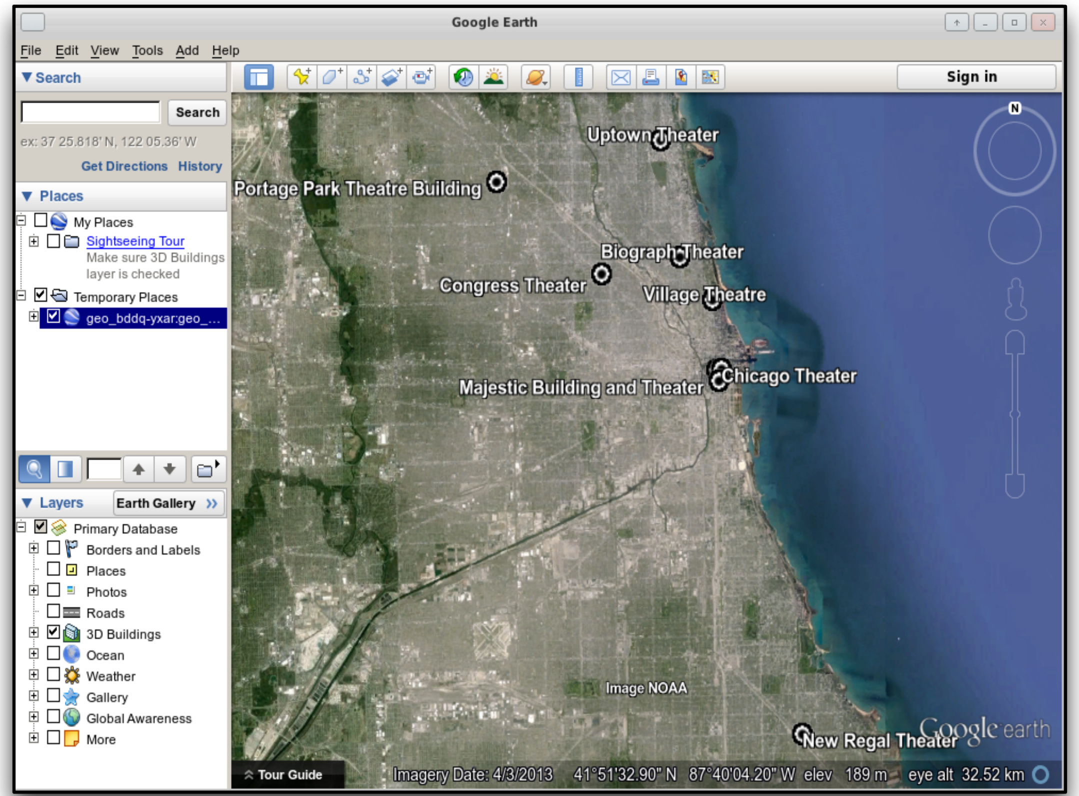
Bob's Machine

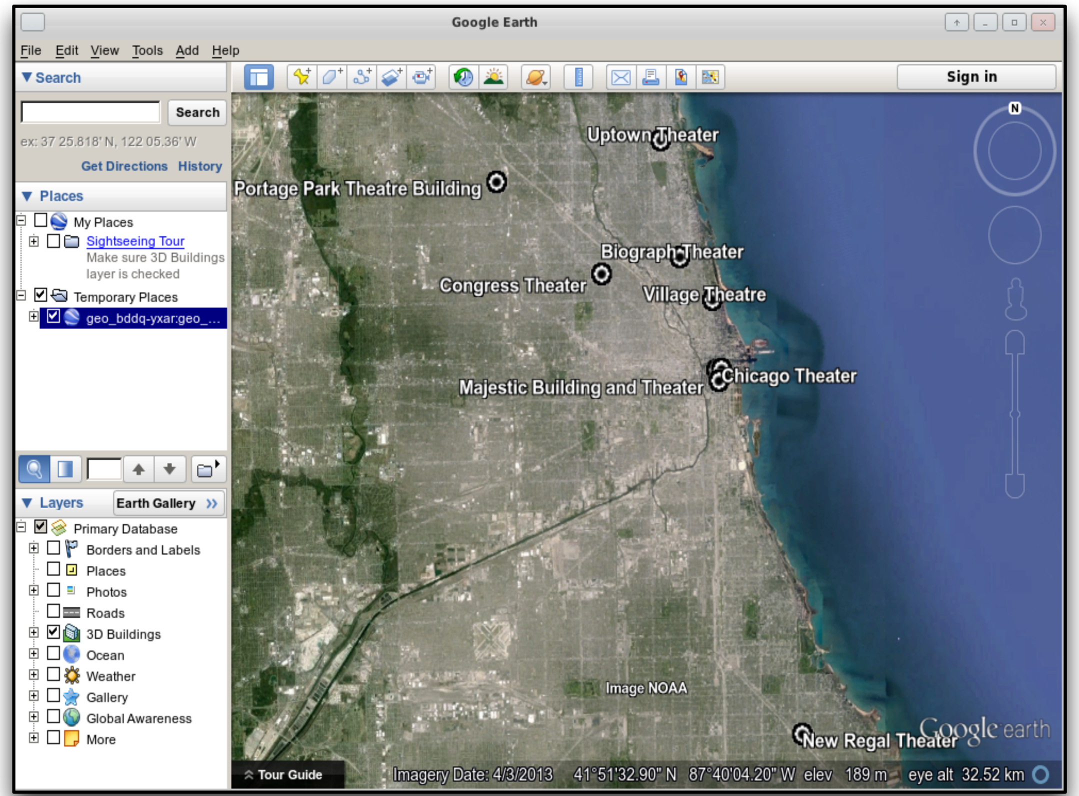


ldv-exec *db-app*

- Redirecting file access
- Redirecting DB access
    - Server-included packages
    - Server-excluded packages

Example





```
Alice:~$ ldv-audit app.sh
Application package created as app-pkg
Alice:~$ ls
app-pkg    app.sh    src    data
Alice:~$echo "Hi Bob, Please find the pkg --Alice" \ l
mutt -s "Sharing DB Application -a "./app-pkg"  \
-- bob-vldb2015@gmail.com
```

```
Bob:~$ ls .
app-pkg
Bob:~$ cd app-pkg
Bob:~$ ls
app.sh      src      data
Bob:~$ldv-exec app.sh
Running app-pkg....
```

Ubuntu 14.04
(Kernel 3.13)
+
Postgres 9.1

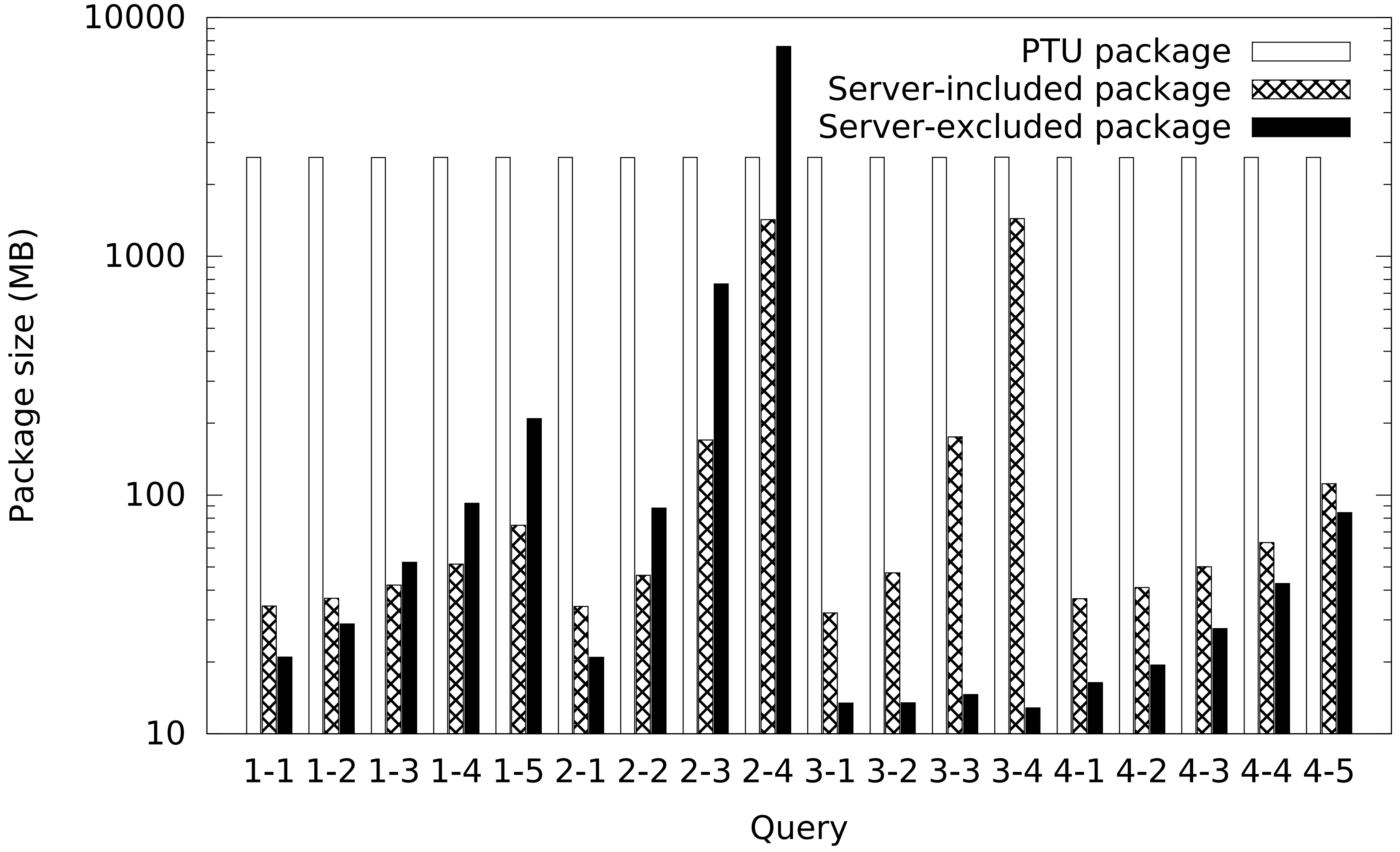
CentOS 6.2
(Kernel 2.6.32)
+
MySQL

LDV Issues

- Monitoring system calls
- Monitoring SQL
- Execution traces
- Relevant DB slices

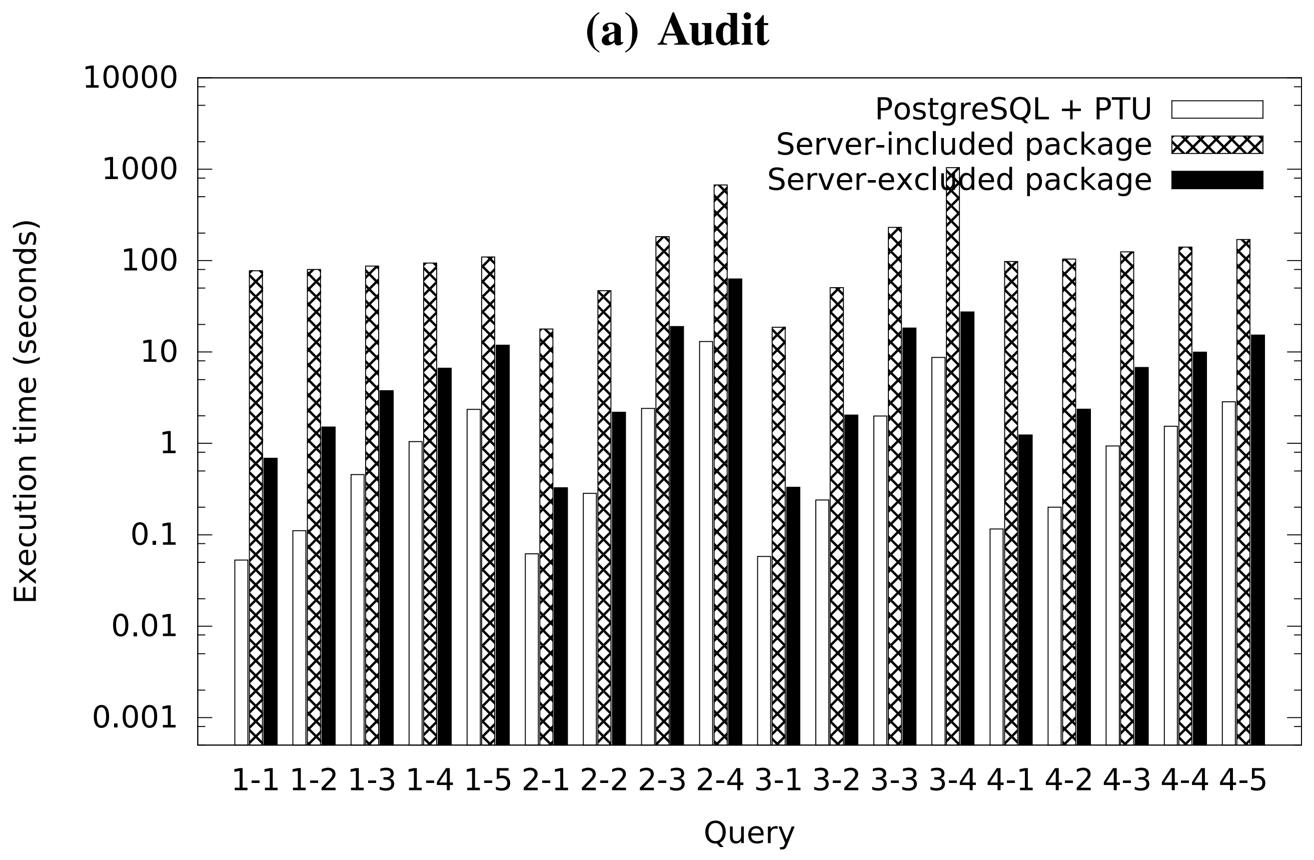
- Redirecting file access
- Server-included packages
- Server-excluded packages
- Redirecting DB access

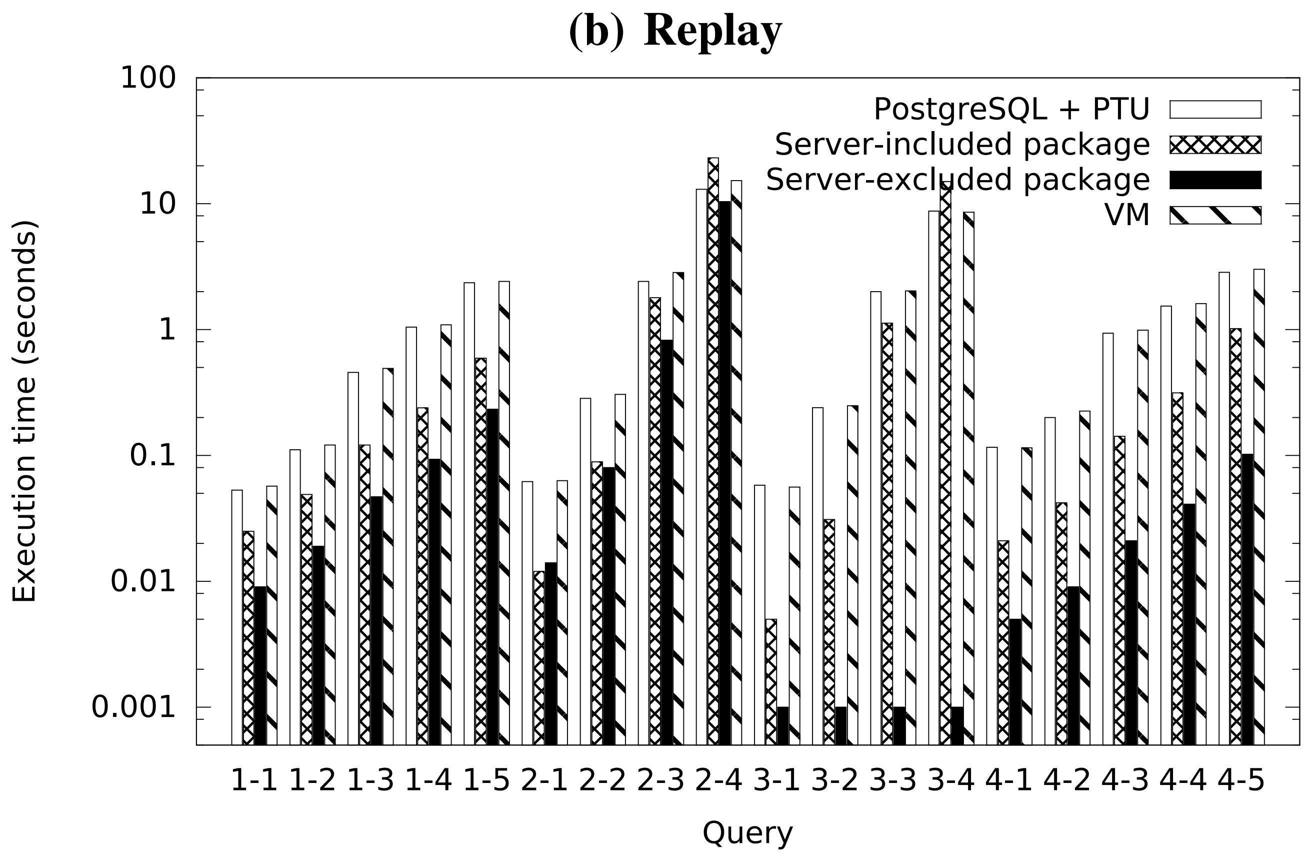
Size Comparison



- LDV packages are significantly smaller than PTU packages when queries have low selectivity
- The VMI is 8.2 GB: 80 times larger than the average LDV package (100MB).

Audit and Replay





LDV amortizes audit cost significantly at replay time

Summary

- LDV permits sharing and repeating DB applications
- LDV combines OS and DB provenance to determine file and DB slices
- LDV creates light-weight virtualized packages based on combined provenance
- Results show LDV is efficient, usable, and general
- LDV at http://github.com/lordpretzel/ldv.git