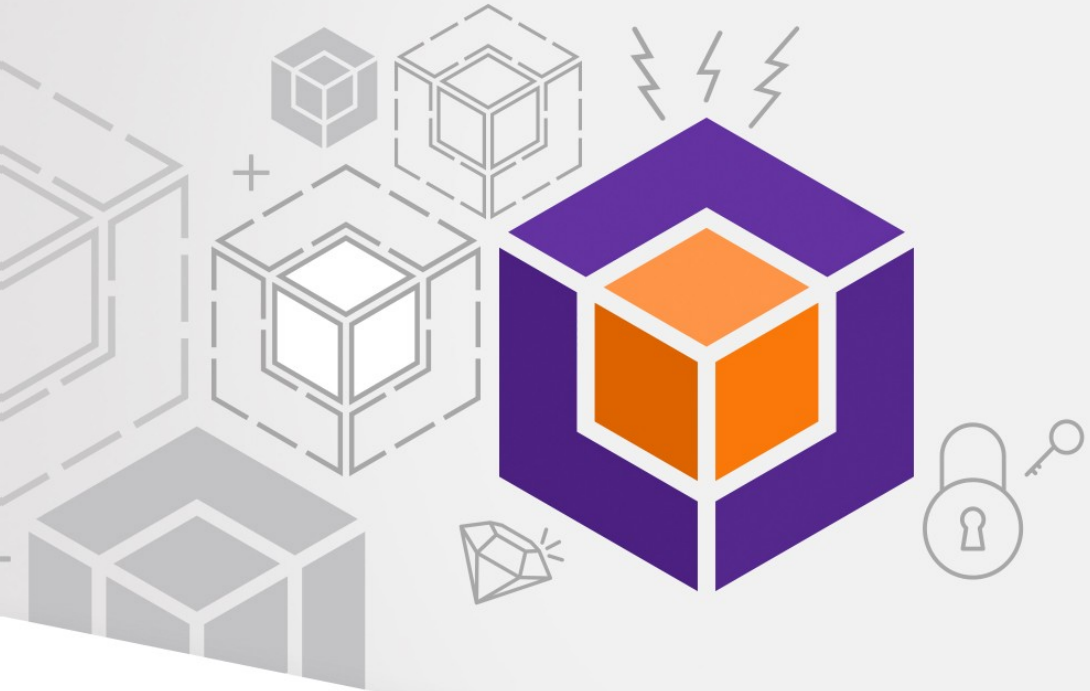
YottaDB

October 2023



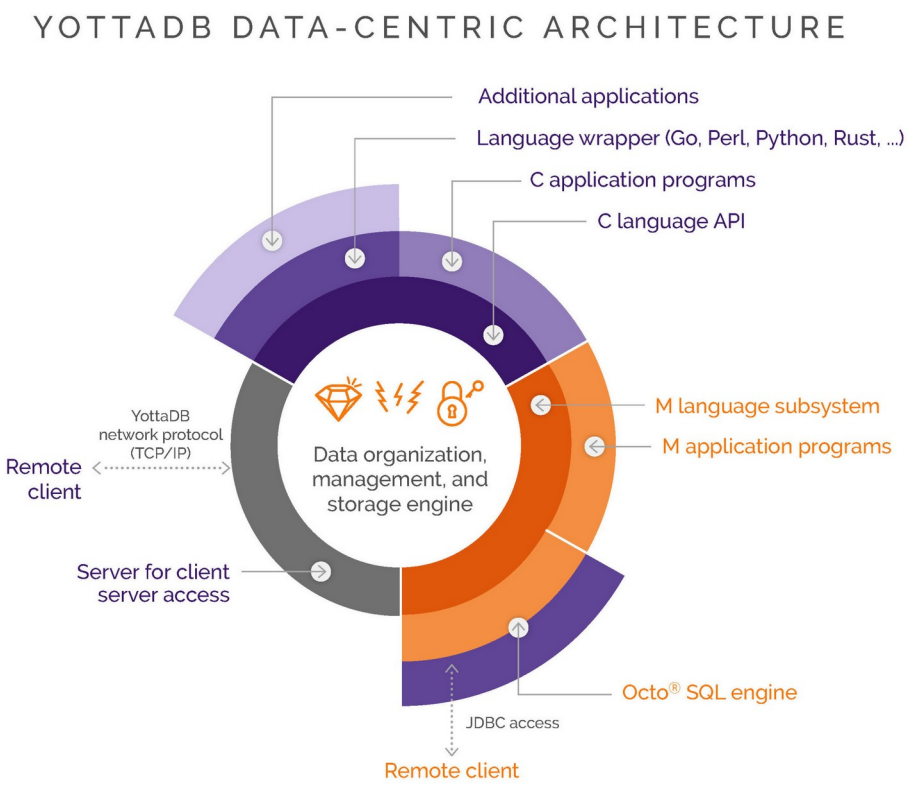
Fun With Forensics

Using SQL to Find Needles in Haystacks

Core Database Technology

- Mature, high performance, hierarchical key-value, *language-agnostic*, NoSQL database whose code base scales up to mission-critical applications like large real-time core-banking and electronic health records, and also *scales down* to run on platforms like the Raspberry Pi Zero, as well as *everything in-between*.
- *Rock Solid. Lightning Fast. Secure. Pick any three.*

Architecture



Supported & Supportable Platforms

YottaDB

	x86_64	AARCH64 (ARM v8)	ARM-HF (ARM v7)
Debian	✓	✓	✓
Ubuntu	✓		
RHEL	✓		
SUSE	✓		

Supportable Platforms

- Debian derivatives: All CPU architectures
- RHEL & SUSE derivatives and other: x86_64
- Build from Source: All CPU architectures on contemporary Linux distributions

Octo – SQL too

- Octo is a SQL database engine whose tables are mapped to YottaDB hierarchical key-value nodes
- Octo runs on YottaDB on 64-bit platforms

Querying Octo

YottaDB

- Terminal session
- YottaDB GUI
- PostgreSQL drivers
  - ODBC driver: Microsoft Excel, PowerBI tools
  - JDBC driver: Dbeaver, Squirrel SQL, SQL Workbench/J
  - Others reported as working, but not tested by us, e.g., Microsoft SSRS, R

YottaDB

Database State Changes

What Changed and When

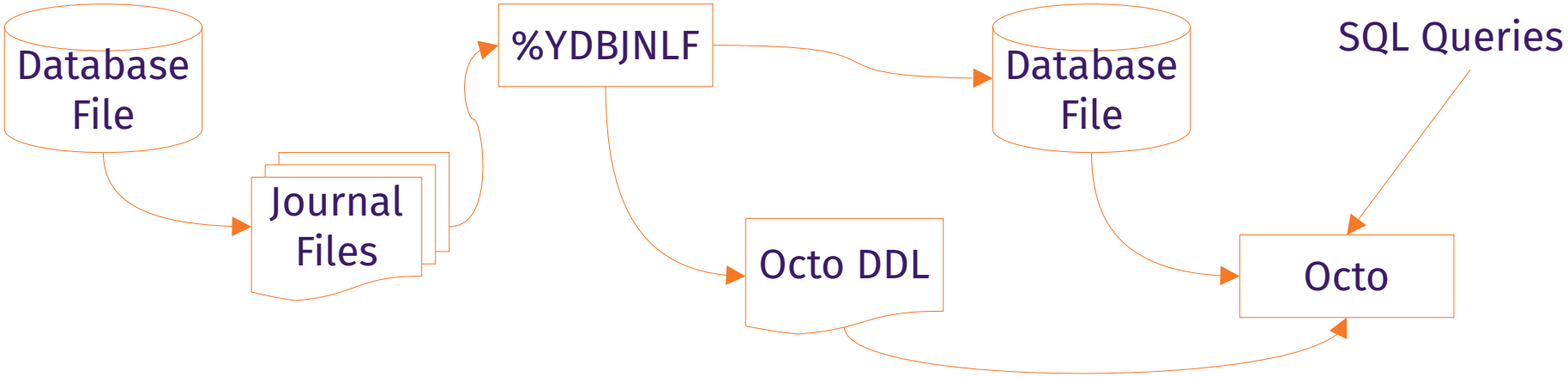
State Machines and State Changes

YottaDB

- Databases are state machines, e.g., a brain transplant has been ordered for Bhaskar
- But the path through state space is also important, e.g., who ordered said brain transplant, and when
  - Journal files capture database state changes
- Yabut… database state changes can outnumber database state
  - How do you find a needle in a haystack?

%YDBJNLF + Octo

- YottaDB databases store large amounts of data
- Octo SQL can query large amounts of data



%YDBJNLF

- Standard YottaDB utility routine
  - https://docs.yottadb.com/ProgrammersGuide/utility.html#ydbjnlf
- `INGEST^%YDBJNLF(`*`jnlfile`*`[,`*`label`*`])` reads `jnlfile` into `^%YDBJNLF`
- `OCTO^%YDBJNLF` produces a DDL that Octo can read
- Automatically creates YDBJNLF region if needed

%YDBJNLF

The %YDBJNLF utility routine loads journal extracts into global variables, allowing software to answer questions such as which process(es) updated a certain global, in what sequence and when; that global variable updates a process made; etc.

Utility Labels

INGEST^%YDBJNLF(jnlfile[,label]) uses MUPIP JOURNAL EXTRACT FORWARD SHOW=ALL FENCES=NONE DETAIL FULL NOVERIFY to extract journal file jnlfile into global variables as described below. Since troubleshooting and forensics may need damaged journal files to be ingested, %YDBJNLF uses the NOVERIFY option.

- If `label` is specified, it is used to identify the extract; otherwise the journal file name `jnlfile` is the identifying label.
- INGEST deletes any existing `^%ydbJNLF*(label)` global variables. Use a unique label for each call to INGEST if the journal file name is not unique, e.g., current

Demo %YDBJNLF + Octo

YottaDB

- *Never call a pool shot with anything other than "Watch this!"*

YottaDB

Syslogs

Computers are where Software lives

Syslogs

YottaDB

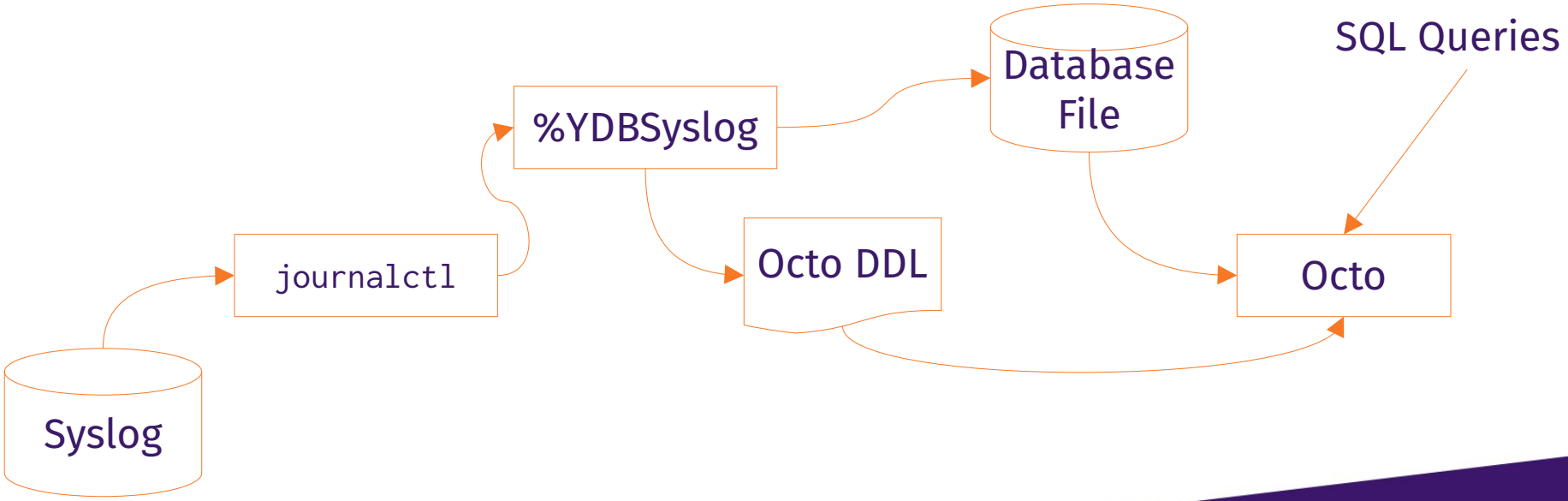
- Computers are bigger state machines than databases
- Networks of computers are bigger yet
- Forensics and troubleshooting often requires looking across multiple computers for events

%YDBSyslog

YottaDB

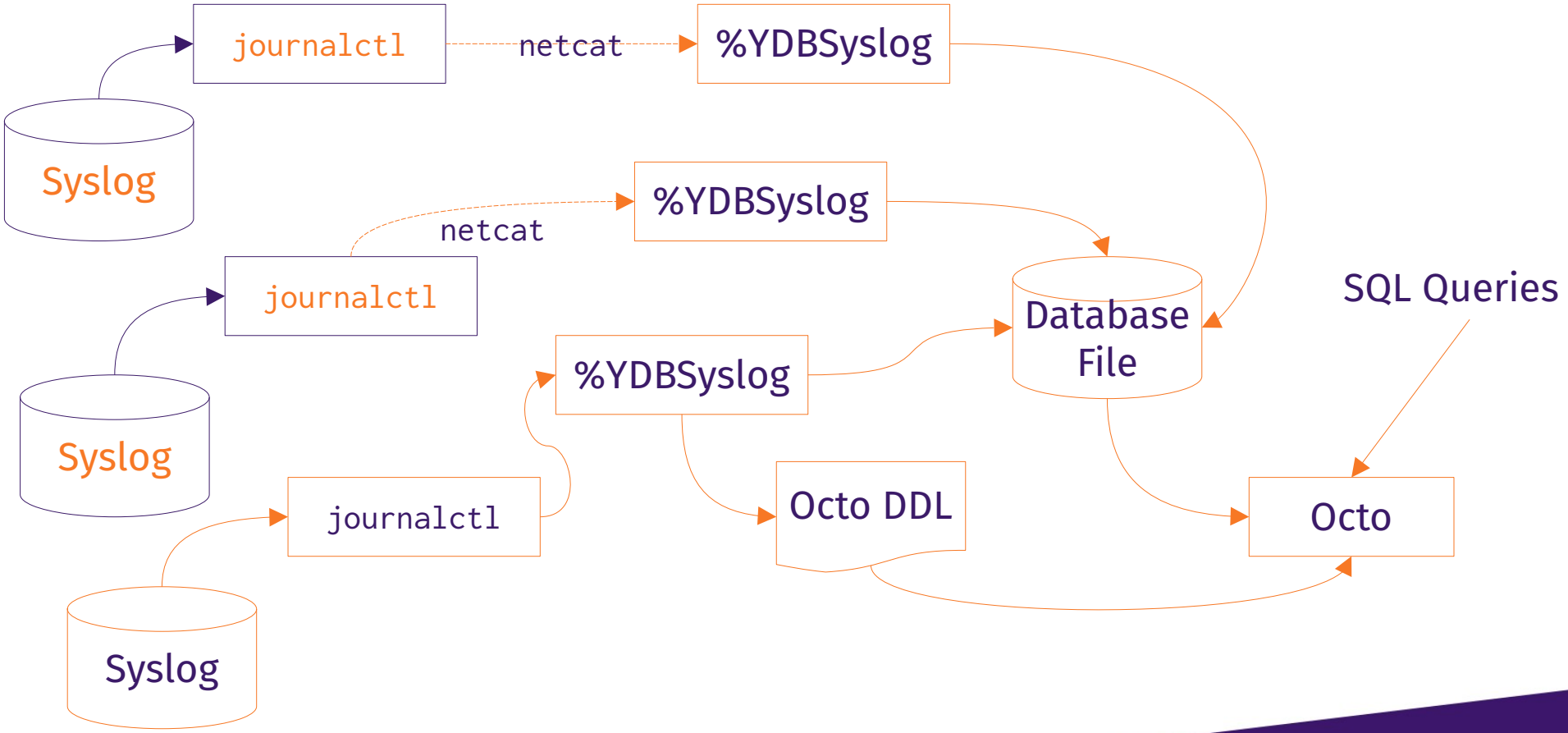
- YottaDB plugin https://gitlab.com/YottaDB/Util/YDBSyslog
- Documented in plugins manual
  https://docs.yottadb.com/Plugins/ydbsyslog.html

%YDBSyslog + Octo



Demo %YDBSYSLOG + Octo

Multiple Machines



YottaDB

Thank You!

K.S. Bhaskar
bhaskar@yottadb.com