Microservices: Get Rid of Your DBA and Send the DB into Burnout

Oracle Code One [DEV5504]

Who am I?

ORACLE Certified Master
Oracle Database 12c Administrator

OakTable®

ORACLE ACE Director

Franck Pachot

Data Engineer at CERN

- Twitter @FranckPachot
- Medium: https://medium.com/@FranckPachot
- Blog Databases at CERN: db-blog.web.cern.ch/

Agenda

1. The database is always a problem
2. Monolithic, Client/Server, n-Tier,… and Micro-Services
3. Get rid of your DBA, Database Burn Out
   … or try to build scalable micro-services

Database is always a problem

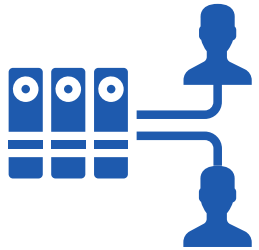
Why do you want to get rid of the database?


Because it is shared


Because it contains persistent data

Because you query it with SQL

Shared



The data is accessed

- by multiple users, doing different things, at the same time

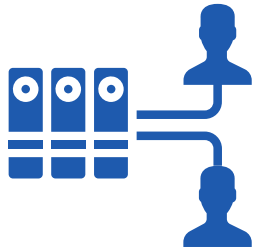
In production,

- one central database, or you need to replicate (with consistency)

In development,

- do you share the dev database, or have one database per developer?
- do you test concurrency scenarios?

Shared



Then comes the idea of micro-services

- With smaller components we have less to share
- Each component has one owner, the only one allowed to refactor

As long as micro-services are defined so that

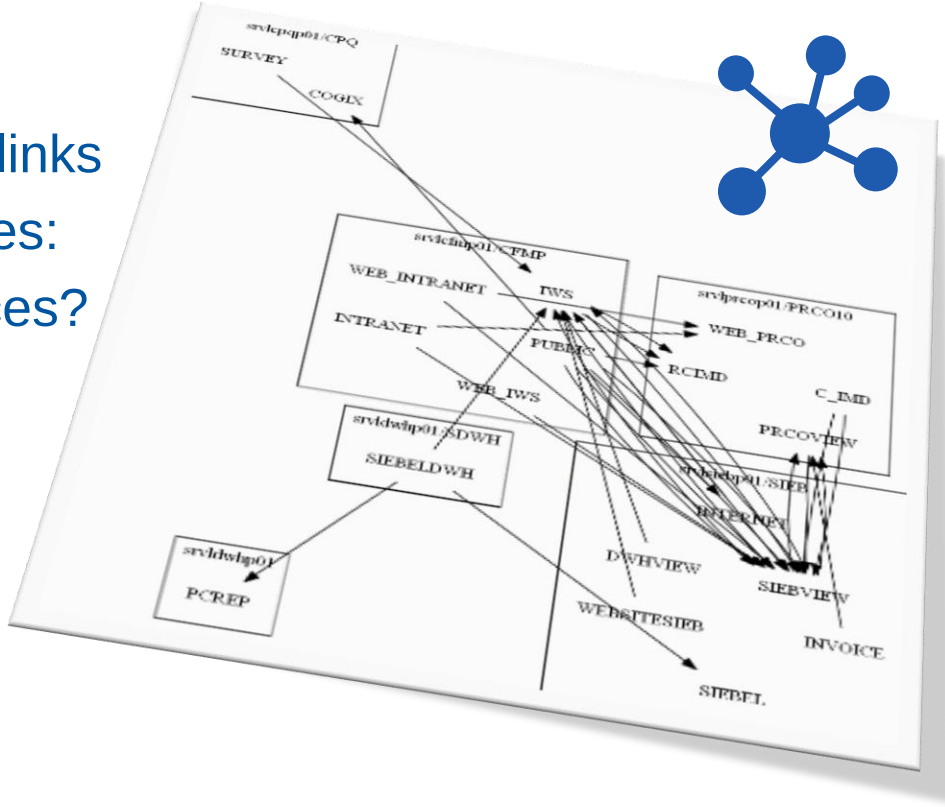
- they do not have data in common
- or can accept an asynchronous replica of data
- Or the interface encapsulated in limited APIs

Are you ready?

Here is a cartography of database links when there were only few databases: Are they ready to go to microservices?

To get it right, you need

- loosely coupled
- no cyclical



Persistent

The application can be restarted, changed
but data must stay forever

This is complex to distribute into multiple services

- You may replicate, but need a single source of truth
- Are you sure to have same protection (backup) for all services?
- CQRS: multiple reads (Query) services, one write service (Command)

You also need consistency

Persistent

You must persist the data and the code to access it

- this is easier with one database
- will you keep all versions of all microservices ?

Unstructured data does not help

- or you need to test all code versions on all data objects

A structured logical layer helps:

- Self-describing data (XML)
- Logical API (SQL views/procedures, DAO, ORM)

SQL

SQL and relational databases are powerful:

- Operates on set of rows (based on math's set theories)
- Normalizes for multi-purpose queries
- Centralizes for consistency, performance, security
- 4th generation language (declarative)

But seems to be too complex for developers and they get back to

- Hierarchical mono-purpose structures (XML, JSON, BSON)
- 3rd generation languages (Java, and even interpreted ones)
- ORM to hide the complexity

SQL

SQL is not only too complex but also

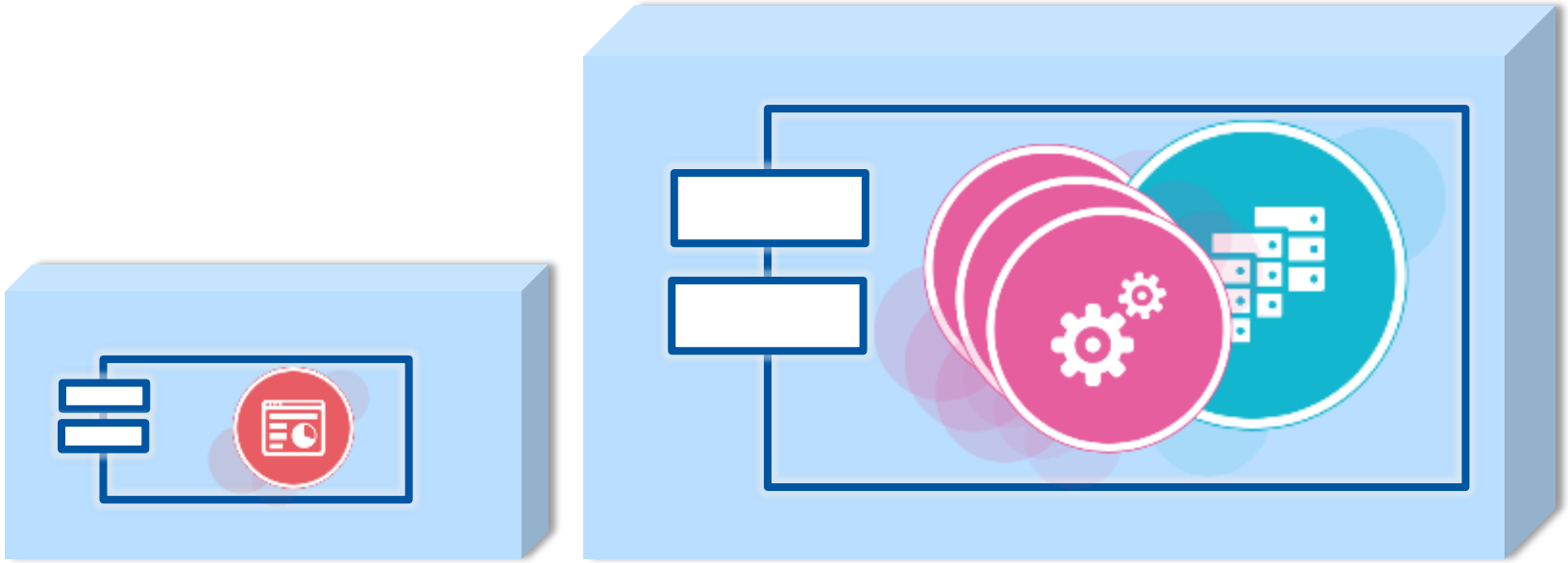
- not portable from one platform to the other (only similar syntax)
- Even worse for procedural extensions (PL/SQL, T-SQL, PL/pgSQL,…)

And then the problem is the cost:

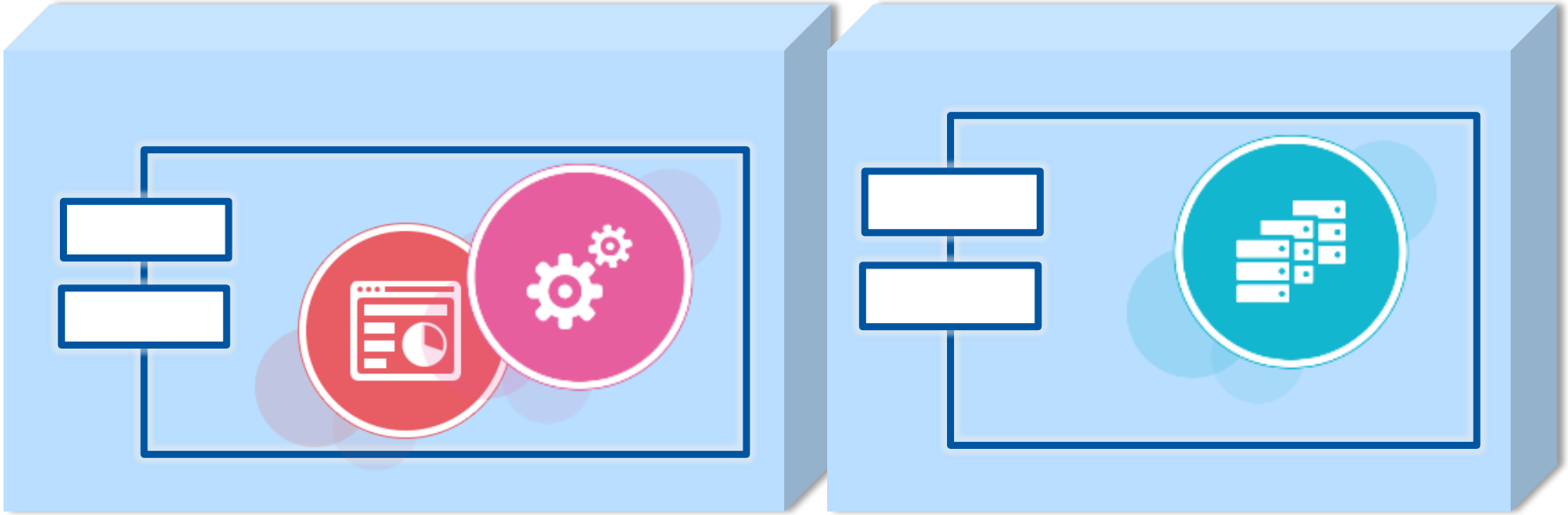
- Powerful RDBMS are expensive
- SQL developers are rare – and expensive

Answer: microservice with easier technology, development offshored

Monolithic servers

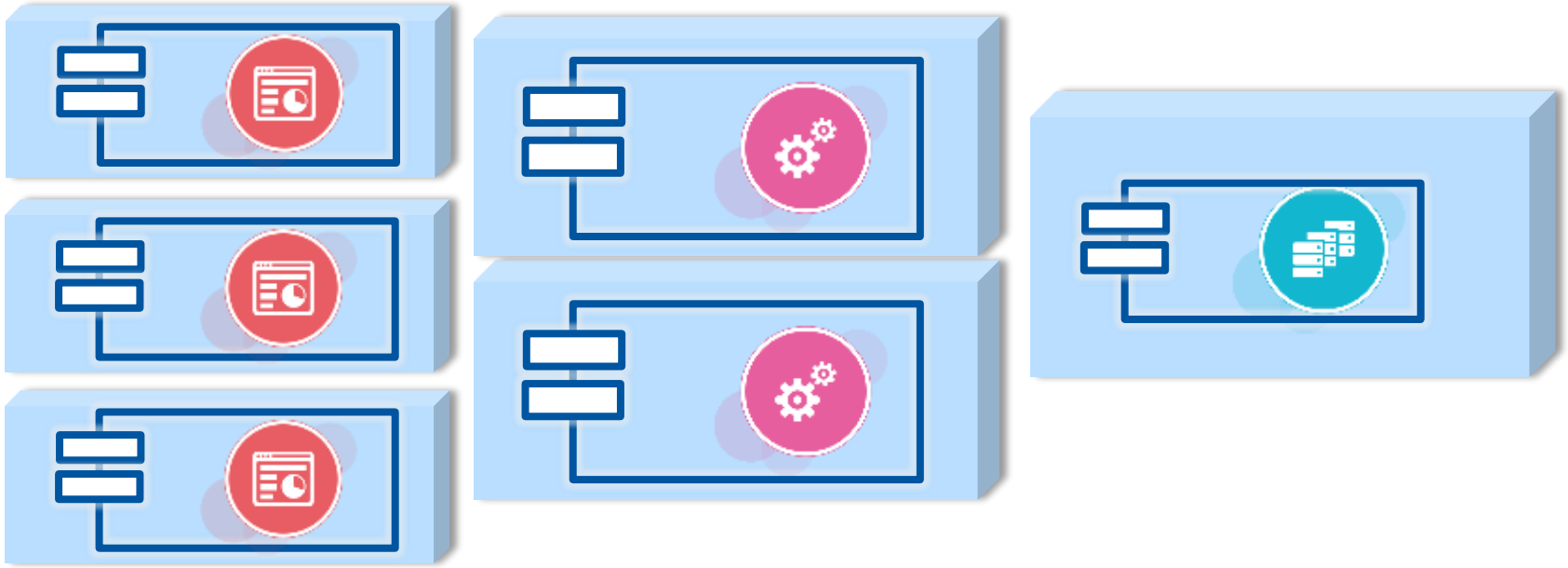


Client server

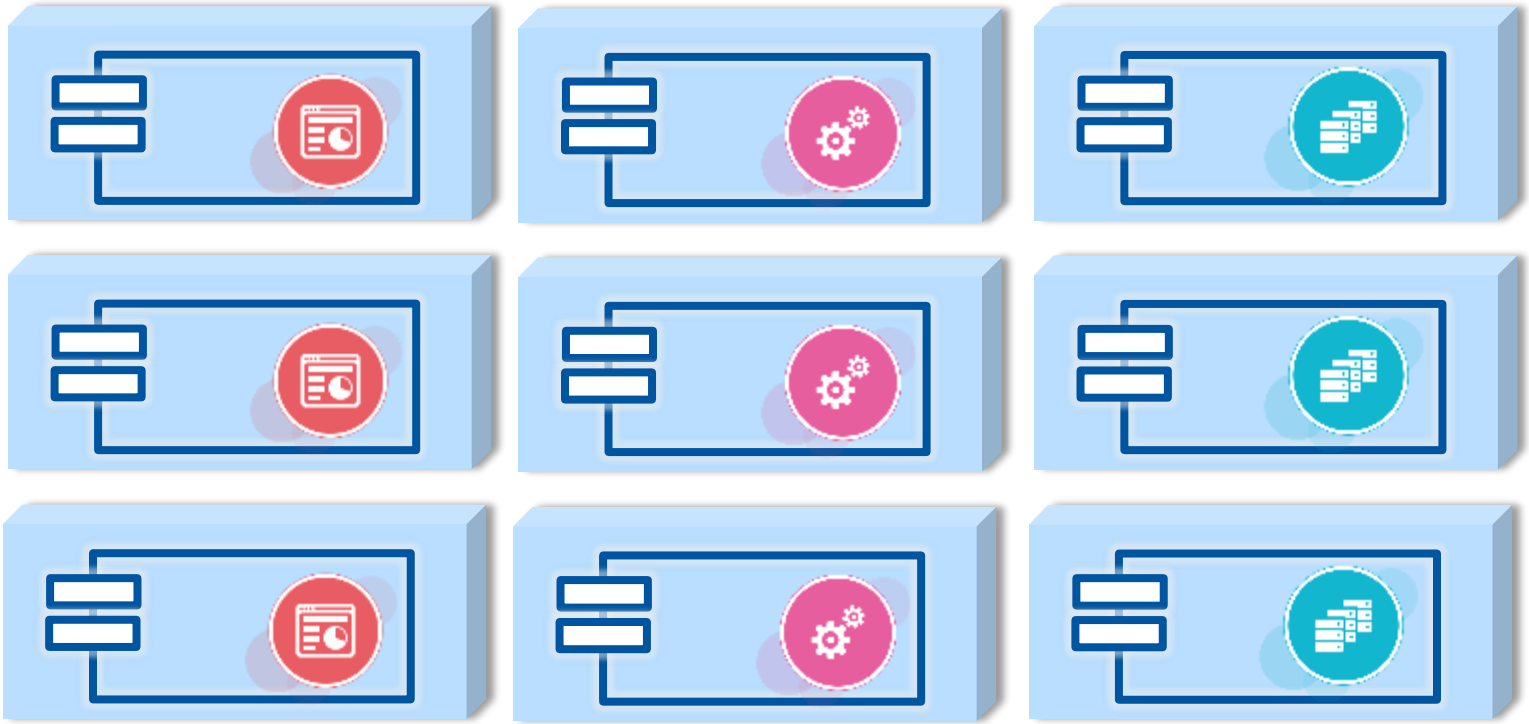


Application server

Application server



Micro-Services



Demo

Looking at the same business logic run from different engines
(doing 500000 transfers between customer account)

- SQL — 5 seconds of CPU
- PL/SQL — 30 seconds of CPU
- JavaScript on client — 2 minutes of CPU (45s in client, 75s in DB)
- JavaScript in DB (MLE) — 1 minute of CPU

Running through different engine, process, machine,… does not scale
and burns more CPU cycles in each tier

Demo

The full content of the live demo is available on the Databases at CERN blog:

http://db-blog.web.cern.ch/blog/franck-pachot/2018-10-odc-appreciation-day-reduce-cpu-usage-running-business-logic-oracle

Context switches

SQL and PL/SQL (or MLE) are running in a different engine

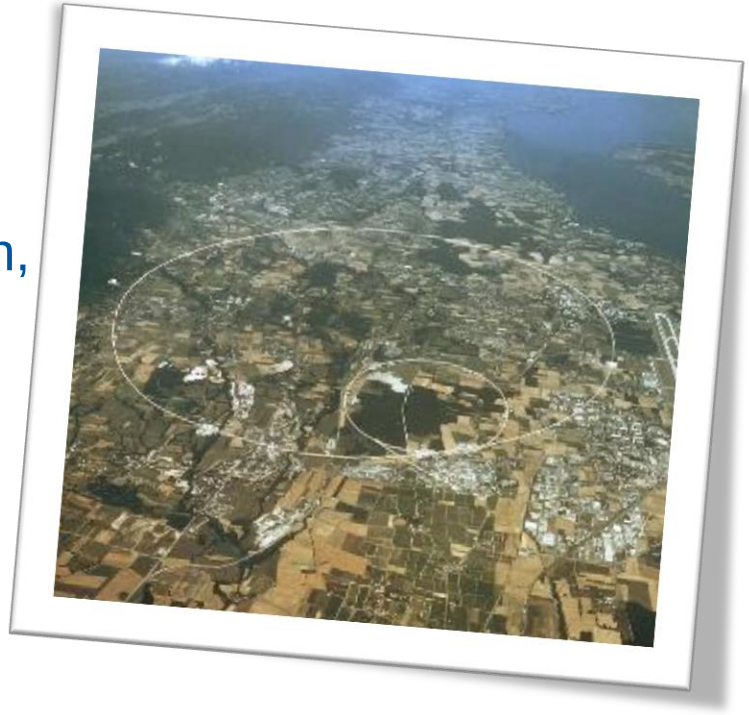
- context switches for each call ⇨ additional CPU cycles

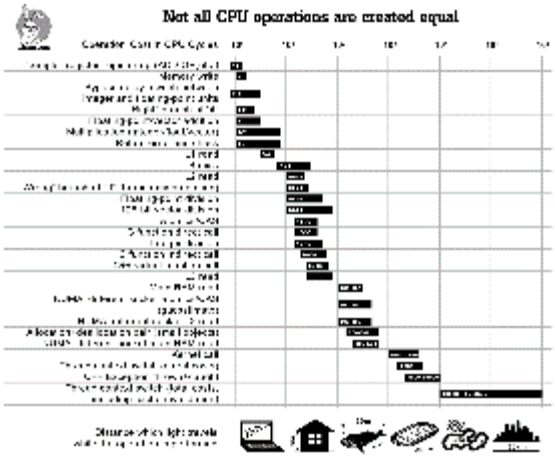
JavaScript in client

- context switch on client when sending the call to the database
- network latency
- context switch on server when receiving the call
- actual work here
- context switch on server when sending the result
- …

Context switches

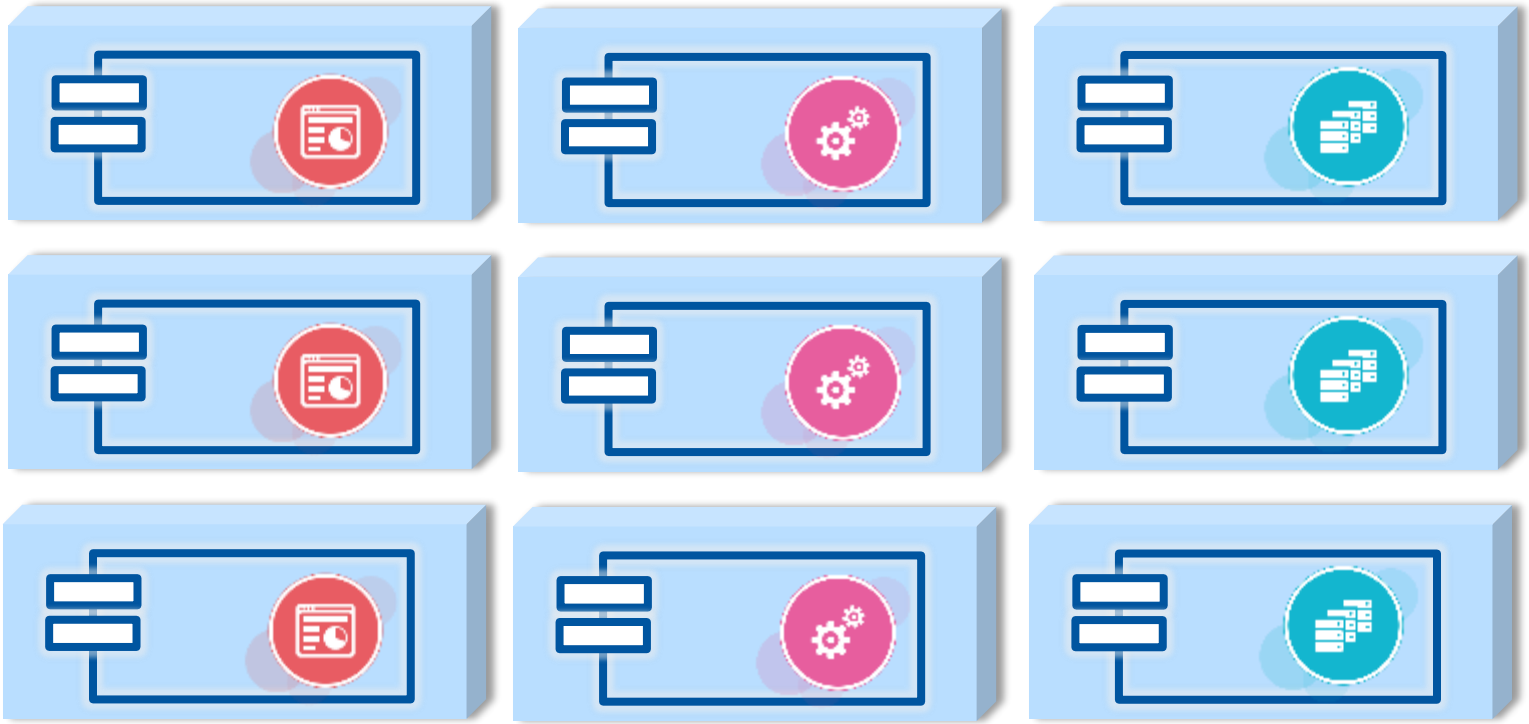
Just to give an idea, during a context switch, the protons can do a complete round of the CERN Large Hadron Collider (27 km)



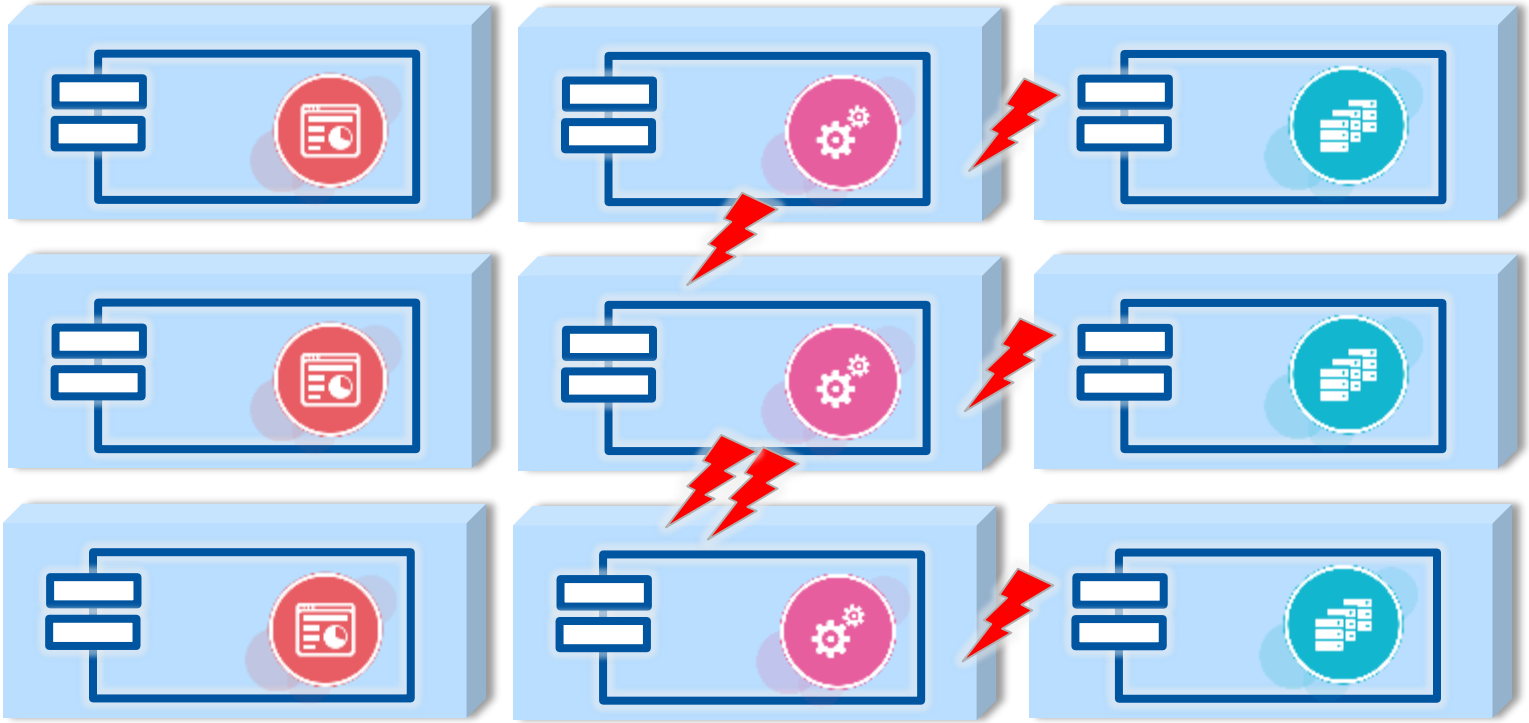


http://ithare.com/infographics-operation-costs-in-cpu-clock-cycles/

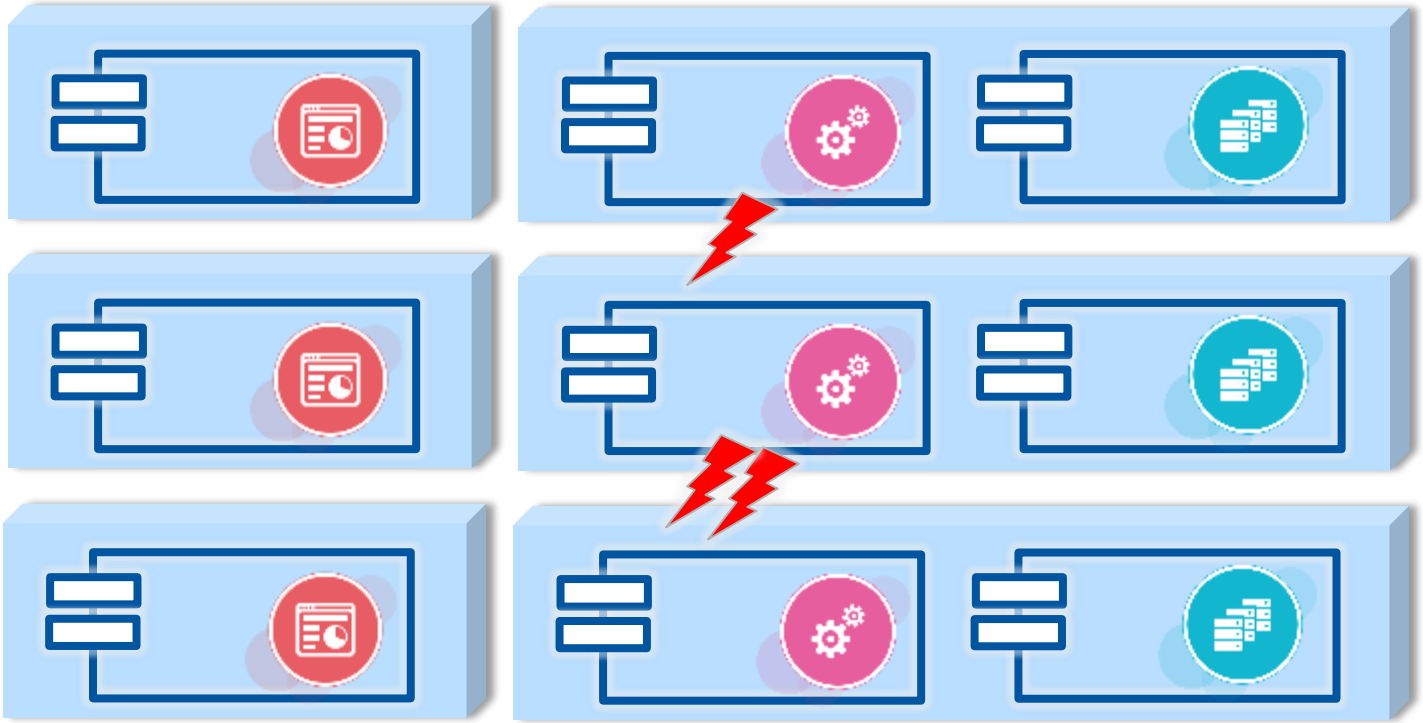
Micro-Services



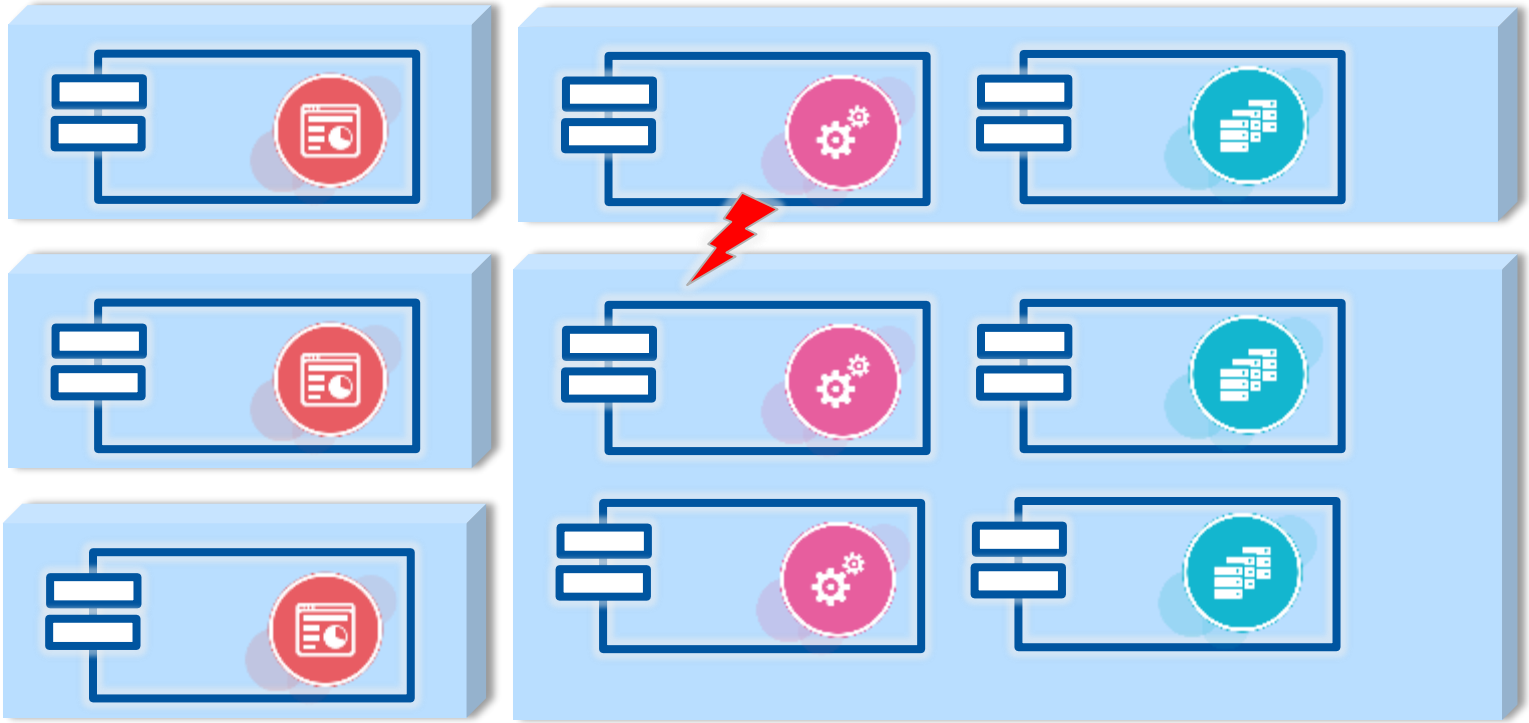
Micro-Services



Micro-Services



Micro-Services



Scalable Micro-Services

How to avoid cross-engine/process/server roundtrips?

Embedded databases

- Berkeley DB, Apache Derby, MongoDB, SQLite,…

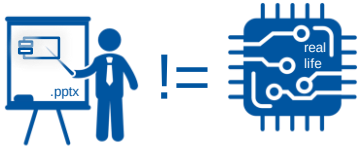
Schemas, Packages, Stored procedures, Views

- provides logical separation (encapsulation, namespace, privileges,…)
- keeps physical co-location (same process, maybe same engine)

• Oracle XE is free (PL/SQL), MLE (future) for Oracle and MySQL

• PostgreSQL can run procedural languages

Core message



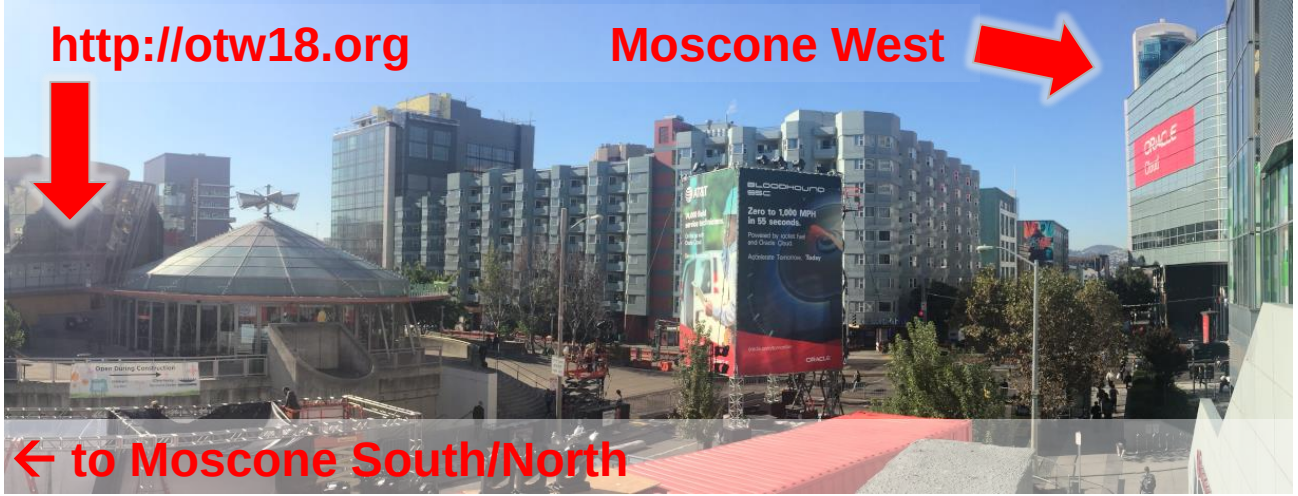
Do not confuse logical with physical architecture

- logical separation is required for organizational purpose
- physical colocation is required for optimal performance

Micro-services must have logical segregation, but they must not add more physical layers with context switches and replication complexity!

- use embedded databases for isolated services
- use views and stored procedures in one database for core data

Enjoy OOW18 and OakTable®



Oak Table World http://otw18.org

- Tuesday 11:00 - 11:50
  Join Methods: NL, HASH, MERGE, & Adaptive in Slow-Motion

Oracle Open World Moscone West 3006

- Wednesday 3:45 p.m. - 4:30 p.m
  Multitenant Security Features Clarify DBA Role in DevOps Cloud