Designing and Implementing for Scale, Distribution & Async

Dr. Jörn Friedrich Dreyer

Solutions Architect

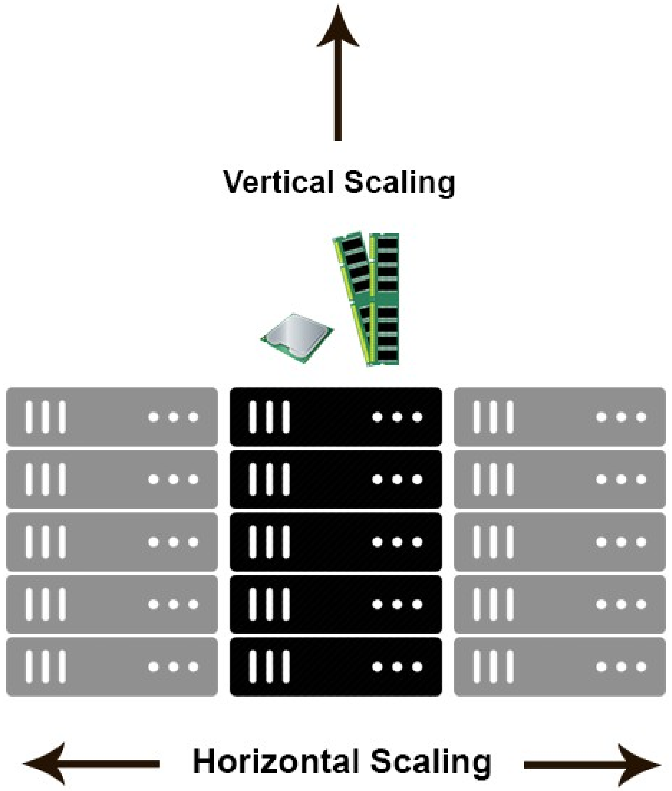
butonic @ twitter, github, ...

Agenda

- Design considerations
  - Vertical and horizontal scalability
  - Fallacies of distributed computing
  - Asynchronicity
- Our implementation focus
  - PHP or GO?
  - ownCloud infinite scale
  - Controlled API evolution with protobuf and gRPC
  - Key takeaway: add LongProcessingResponse to cs3 apis!

It’s all theory …

Scalability



source: https://stackoverflow.com/a/11715598

Fallacies of distributed computing

1. The network is reliable.
2. Latency is zero.
3. Bandwidth is infinite.
4. The network is secure.
5. Topology doesn't change.
6. There is one administrator.
7. Transport cost is zero.
8. The network is homogeneous.

source: https://www.datawire.io/using-fallacies-of-distributed-computing-to-build-resilient-microservices/

Asynchronicity

- Async IO
  - your current thread is not blocked until the remote service has responded
  - mechanisms for working with non-blocking IO are callbacks, futures, or streams
- Asynchronous protocols
  - HTTP is a synchronous protocol: the client issues a request and waits for a response
  - message passing: as a sender, you usually don’t wait for a response
- Asynchronous service integration
  - Do not communicate with other services during your own service’s request/response cycle.
- Self-Contained Systems (SCS)
  - Whenever feasible, “integration” between two self-contained systems should happen in the UI

source: https://www.innoq.com/de/blog/why-restful-communication-between-microservices-can-be-perfectly-fine/

Making it real ...

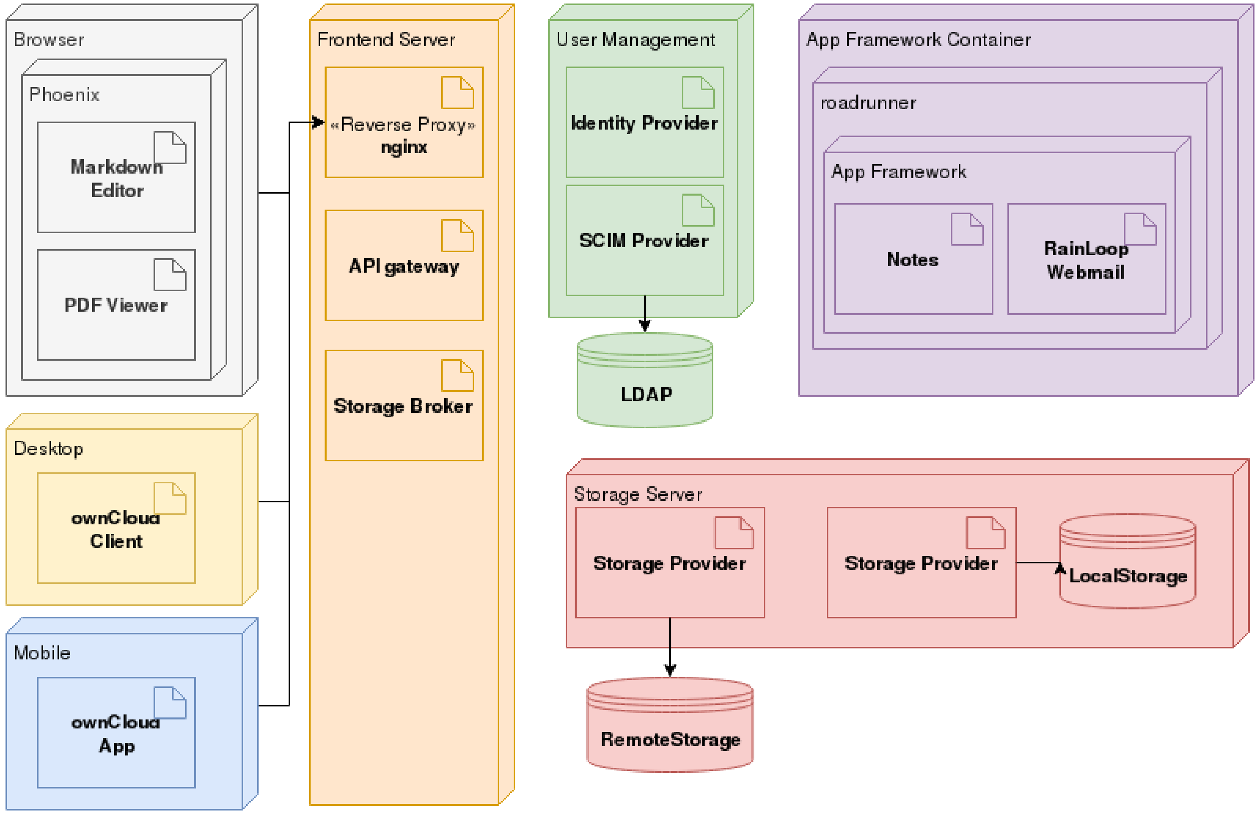
PHP or GO?

- PHP
  - Interpreted (well cached)
  - Request driven
  - OS managed threads
  - Yield allows some thread concurrency
  - fibers are being worked on as ext-async
  - Blocking IO
    - All async efforts (ReactPHP, amp, or swoole) reimplement mysql and redis access
  - Ecosystem built with blocking IO in mind

- golang
  - Compiled language (short build times)
  - Long running services
  - Userspace managed coroutines
  - Async IO is an integral part
    - Storage IO is blocking, though
  - The community seems appalled with non thread safe libraries like libsmbclient

- golang is more mature for async io

ownCloud infinite scale



Controlled API evolution with protobuf and gRPC

- protobuf is an IDL
- Client and server stubs are generated
  - C++, Java, Python, Go, Ruby, Objective-C, C#, JavaScript
  - third-party implementations for C, Perl, PHP, R, Scala, Swift and Julia
  - Binary protocol & HTTP/2
- Even browser based clients can use it
- Supports an evolving schema!
- CS3 APIs feature request #1:
  allow async requests by adding a **`LongRunningResponse`**

```
syntax = "proto3";

message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
}

service SearchService {
  rpc Search (SearchRequest)
    returns (SearchResponse);
}
```

Thank You

owncloud.com

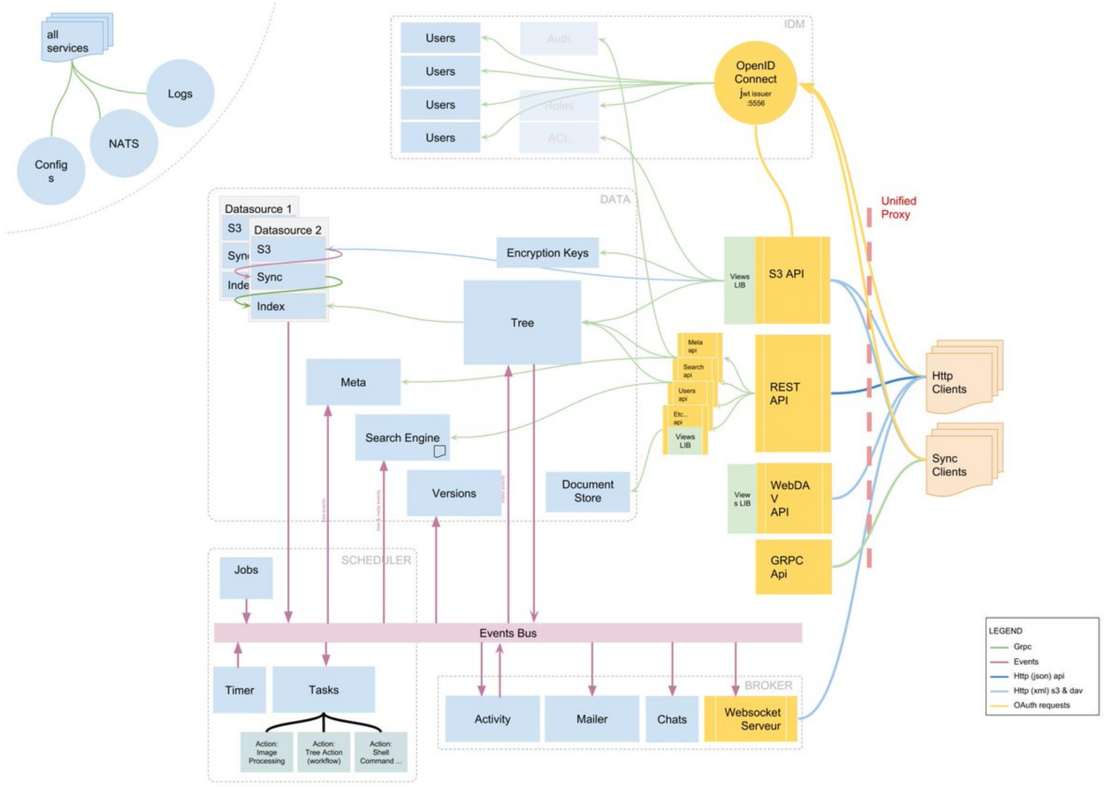
facebook.com/ownCloud

@ownCloud

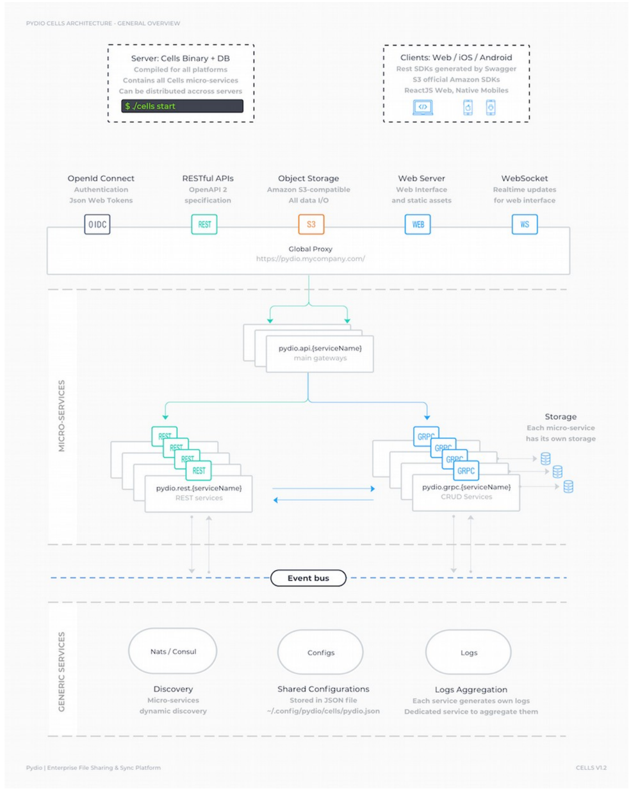
Rathsbergstr. 17, 90411 Nürnberg

+49 911 14888690

Pydio Cells

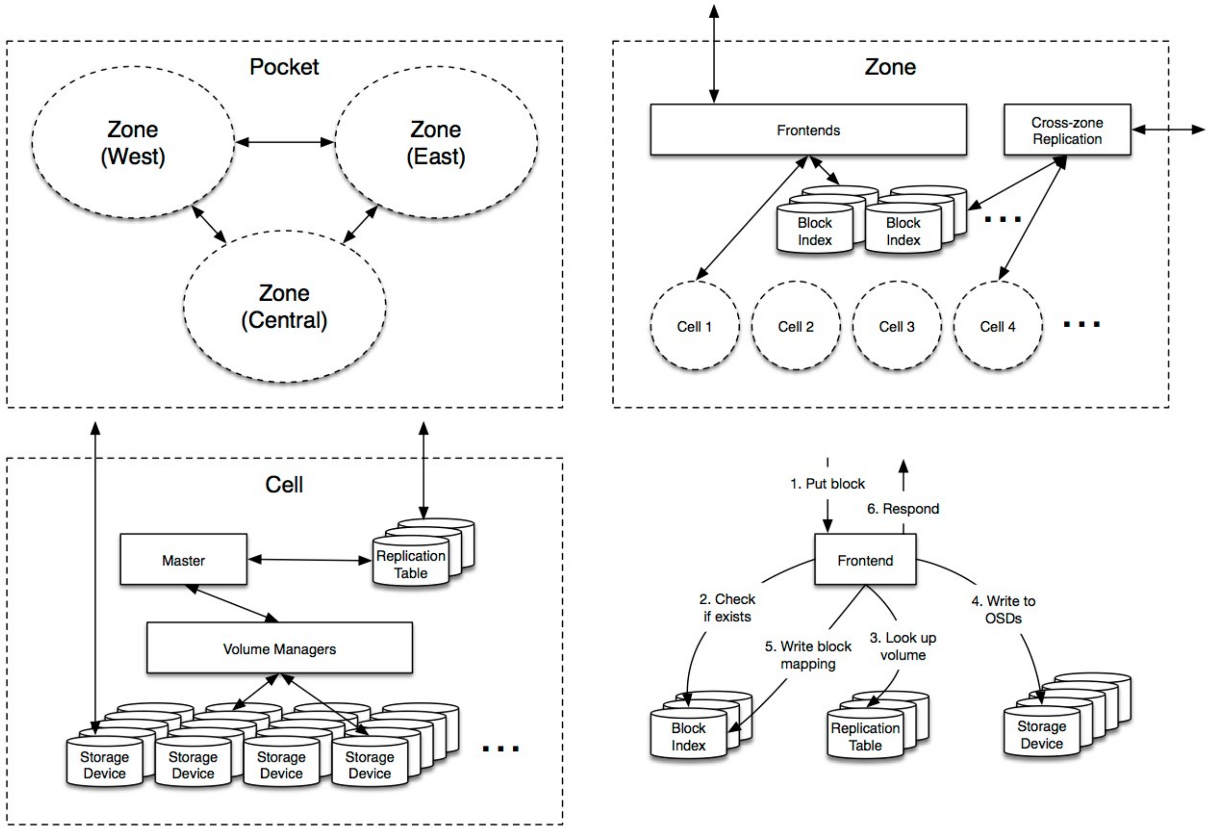


all services
Logs
NATS
Config s
IDM
Users
Users
Users
Users
Auth
Roles
ACL
OpenID Connect
jwt issuer :5556
DATA
Datasource 1
S3
Sync
Index
Datasource 2
S3
Sync
Index
Encryption Keys
Tree
Meta
Search Engine
Versions
Document Store
Views LIB
S3 API
Meta api
Search api
Users api
Etc... api
Views LIB
REST API
View s LIB
WebDAV API
GRPC Api
Unified Proxy
Http Clients
Sync Clients
SCHEDULER
Jobs
Events Bus
Timer
Tasks
Action: Image Processing
Action: Tree Action (workflow)
Action: Shell Command ...
BROKER
Activity
Mailer
Chats
Websocket Serveur
LEGEND
Grpc
Events
Http (json) api
Http (xml) s3 & dav
OAuth requests

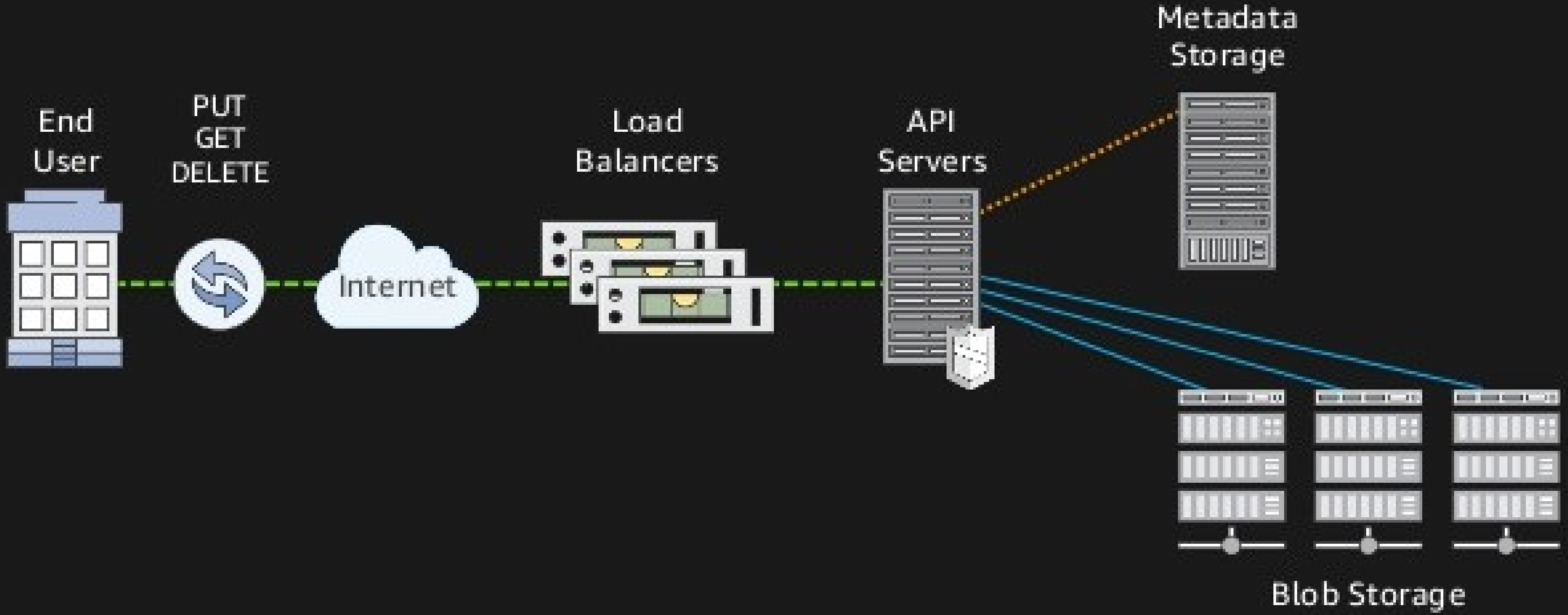


PYDIO CELLS ARCHITECTURE - GENERAL OVERVIEW
Server: Cells Binary + DB
Compiled for all platforms
Contains all Cells micro-services
Can be distributed accross servers
$ ./cells start
Clients: Web / iOS / Android
Rest SDKs generated by Swagger
S3 official Amazon SDKs
ReactJS Web, Native Mobiles
OpenId Connect
Authentication
Json Web Tokens
RESTful APIs
OpenAPI 2
specification
Object Storage
Amazon S3-compatible
All data I/O
Web Server
Web Interface
and static assets
WebSocket
Realtime updates
for web interface
OIDC
REST
S3
WEB
WS
Global Proxy
https://pydio.mycompany.com/
MICRO-SERVICES
pydio.api.{serviceName}
main gateways
REST
pydio.rest.{serviceName}
REST services
GRPC
pydio.grpc.{serviceName}
CRUD Services
Storage
Each micro-service
has its own storage
Event bus
GENERIC SERVICES
Nats / Consul
Discovery
Micro-services
dynamic discovery
Configs
Shared Configurations
Stored in JSON file
~/.config/pydio/cells/pydio.json
Logs
Logs Aggregation
Each service generates own logs
Dedicated service to aggregate them
Pydio | Enterprise File Sharing & Sync Platform
CELLS V1.2

Dropbox Magic Pocket



Amazon S3 Architecture


End User
PUT
GET
DELETE
Internet
Load Balancers
API Servers
Metadata Storage
Blob Storage