为什么要使用 BukkitHTTP

BukkitHTTP Dev Team

Intro：什么是On-Premises商业模式

- 假定A、B、C等多个学校各需要一套教务系统。
- 什么是传统商业模式？
  - 开发并在自己的服务器上运行教务平台X
    - 为A、B、C提供校级账号
    - 校级账号可以添加学生、老师账号
- 什么是On-Premises商业模式？
  - 开发教务系统X
    - A、B、C自行通过X搭建教务平台
    - A、B、C的师生访问本校的平台

Intro：On-Premises商业模式的特点

传统网络商业模式（出售平台服务）

>数据会离开使用者平台，因为合规要求难以ToB

>需要相关计算设施，增加后期维护投入成本

>如果客户数量较多，即使单个用户的体量不大，也需要高并发架构，拖延研发周期，增加研发成本

>如果因为运维失误，数据外泄/丢失，平台会成为责任主体，为后期埋雷

>不具备物理隔离可能性，不能用于涉密等项目

>难以定价，因为买断制会导致运行成本逐日增加，不可持续；而订阅制难以吸引大体量用户

On-Premises

>数据不离开使用者平台，满足合规性要求

>硬件环境由使用者提供，只需要提供技术服务

>原理上完成分布式计算，本质上属于微服务

>即使出现运维失误，开发方也没有责任，还可以通过提供辅助运维服务获取额外利润

>自行部署，可以满足物理隔离等要求，可以是涉密项目

>易于定价，0维护成本消除了买断制的不可持续性；订阅制可以获得持续收益

传统框架实现On-Premises的挑战

- 部署时的客户流失
  - 传统框架部署复杂度较高，如SpringBoot的经典部署流程十分繁琐（如下），很有可能导致用户因为难以完成而转向其他解决方案
    - 1 部署MySQL
    - 2 配置数据库用户与表
    - 3导入初始数据库文件
    - 4 修改程序配置
    - …
- 续约时的客户流失
  - 与传统框架的安装门槛高相对的是，传统框架平替门槛低，用户在续约时可能替换为其他同类解决方案
    - 很多软件甚至直接制作了同类软件数据导入功能，如GitLab可以导入Gitea的数据库

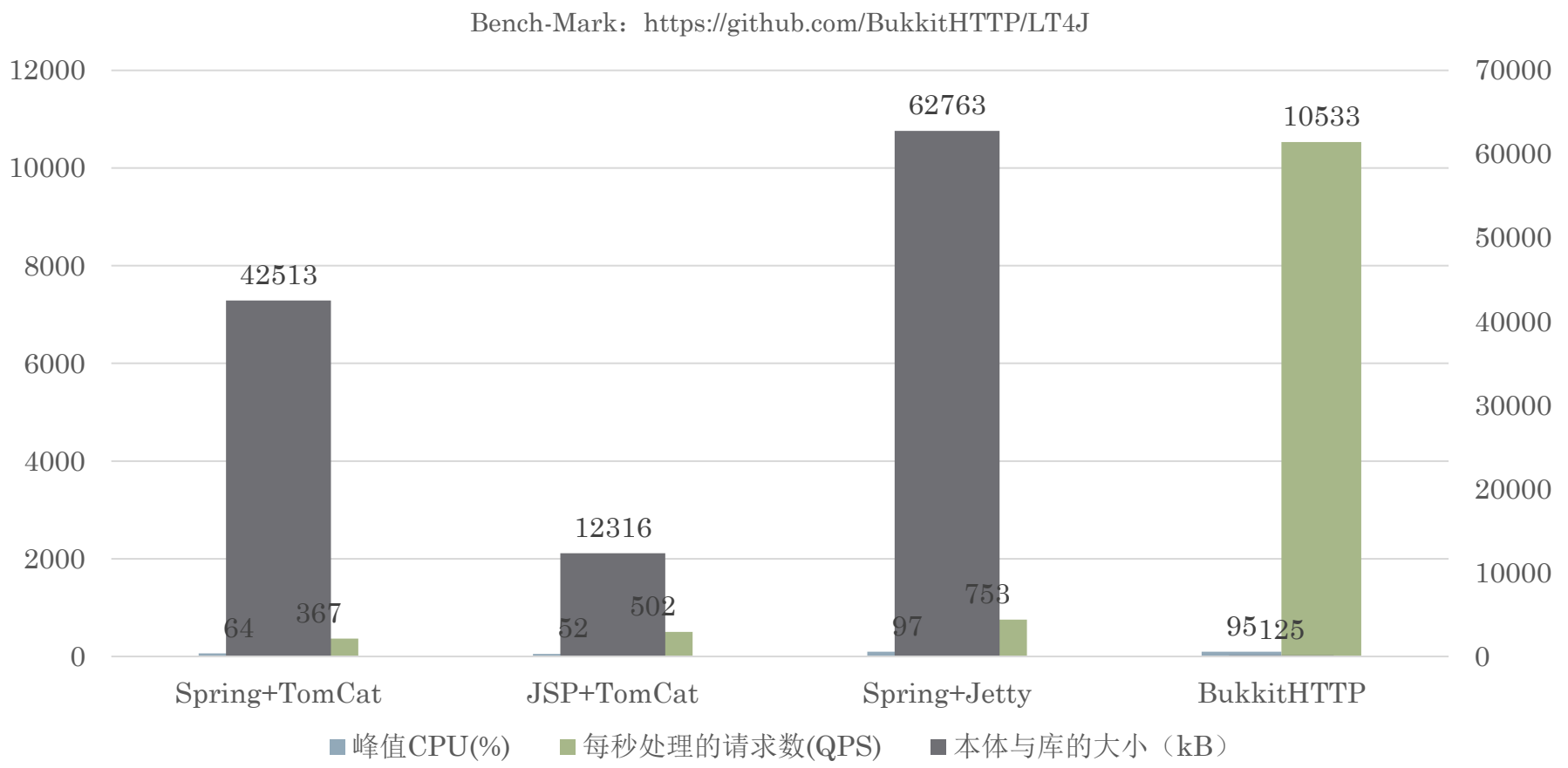
BukkitHTTP：挑战的应对者

- 简单的部署过程：
  - 填写注册码后即可直接运行。无论有无数据库，都不需要配置任何其他内容
- 内置的认证系统：
  - 内置与硬件绑定的离线模块授权认证系统。兼顾对离线闭环的合规性要求、对代码保护（防逆向、破解）的强度要求和使用、开发的简便性
- 独特的数据库格式：
  - 易于导出、备份：可以轻易的将数据作为表格导出或者作为数据库文件整个备份，大幅降低使用和管理难度
  - 高性能：没有不必要的跨进程通信和值传递，高并发时性能不输MySQL
  - 易于开发：Object直接作为DTO使用，缩短开发周期
  - 安全：数据库文件加密存储，降低泄漏后带来的风险；没有模块源码难以使用

统一的跨平台工具链

- 运行于每一个终端
  - 我们不滥用Native，因此可以在BSD、Solaris等不常见环境上运行
- 统一每一个接口
  - Db即Dto即Object即JSON。没有不必要的拷贝与命名。
- 提供每一个工具
  - 包含数据库，日志，验证码，Json，控制台输入&更多

卓越的体积、CPU利用率与速度



不打破您的开发团队

- 极低的学习与迁移成本
  - 使用Java原生语法
  - 不定义Annotations
  - 不需要IDE插件
  - 保留每个人熟悉的开发环境
- 周到的兼容性
  - 正确处理getter/setter风格与public风格
  - 不同的模块间有类隔离
  - 同时胜任前后端分离与全栈开发

保障您的运行安全

- 规避来自依赖的威胁
  - 因为使用Log4j，运行SpringBoot的所有服务器都存在过任意命令执行漏洞
    - CVE-2021-44228 与 CVE-2021-45046
  - 因为使用FastJson，运行MyBatis的大部分服务器都存在过任意命令执行漏洞
    - CVE-2022-25845
  - …
- 规避来自攻击的威胁
  - 内置NanoFirewall，自带筛选、识别并屏蔽潜在的恶意流量
  - 原生的数据库设计，掐断SQL注入的每一丝可能
  - 使用统一管理的Session而不是Cookie，会话具体内容不离开服务端

组织您的运维团队

- 数据库物理访问与权限访问隔离
  - 文件化数据库，备份数据库、回档数据库等行为不受影响，甚至更加轻松
  - 加密型数据库，避免运维团队有意或无意修改
- 安全的权限与角色
  - 在有合适的代码审计的情况下，实现去中心化的产品生命周期
  - 类似于双钥匙机制，保证运维团队易于组织

加速您的产品上线

- 隔离的ClassLoader环境赋能协同开发
  - 冲突类兼容性
    - 通过修改双亲委派类加载顺序为本地优先，不同的模块可以有完全相同名称的类而互不干扰，每个开发者只需要专心于自己的代码而留给BukkitHTTP自动解决代码冲突
  - 后期路由修改
    - 每一个模块的路由都可以在不修改源代码的情况下被调整，减轻架构工程师的负担
- 从开发到生产无缝对接
  - 易于备份的数据库
    - 将模块拷贝到本地即为开发环境，拷贝到云端即为开发环境，数据隔离而不需要额外配置
  - 易于管理的生产环境
    - 热卸载与热加载模块，即使模块出错也可以在不重启且不影响其他模块运行的情况下修复，大幅降低试错成本

感谢您的查看

BukkitHTTP/BukkitHTTP: Backend, Redefined. (github.com)