小规模实验室
自动化和计算系统的建设

罗嘉宏

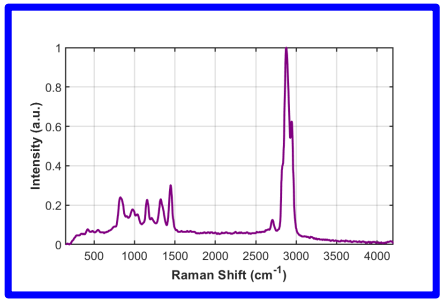
中国科学技术大学化学物理系

luojh@mail.ustc.edu.cn

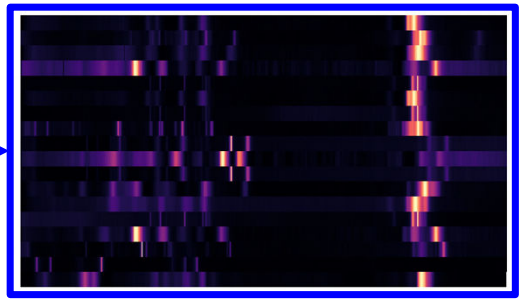
目录

1. 概述

- **自动化高效率**的灵活**计算**、**数据处理**需求
  - 例如，对大量（10^6？）图像和谱学数据进行特征提取



Peak #	Center	Height	Width
1	2900	1.0	100
2	1450	0.3	20
…			



- 硬件设施
- 配套软件（计算、管理）
- 业务逻辑

2. 系统构成

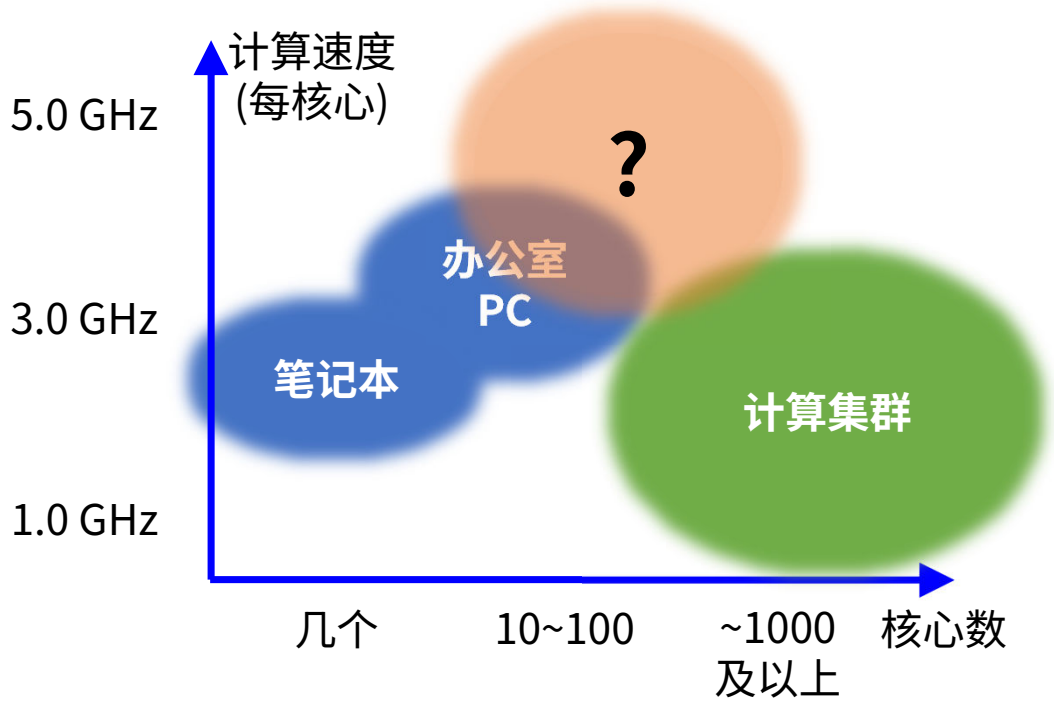
- **计算**
  - **小计算量**的任务占主体，大计算量的上超算
  - **精细高效**地分配好有限的计算资源（精确到 CPU 时间）
  - **隔离**、**一致**的运行环境
  - 自主设计程序的源代码管理、CI/CD
- **存储**
  - 大量**实验数据**从不同的设备上采集
  - 远、近，快、慢均有
  - **可靠**、**中心**存储，支持自动化程序的数据采集
  - 分层级、速度组织存储器

2. 系统构成

- **网络**
  - 安全可靠的**内部高速互联**（≥1Gbps）
  - 为特定的（有需求的）线路提高带宽
  - 实验中心之间有网络访问
- **计算系统**
  - 设备环境监测
  - 计算节点在线检查
  - 视频监控
  - 电力供应
  - 系统维护

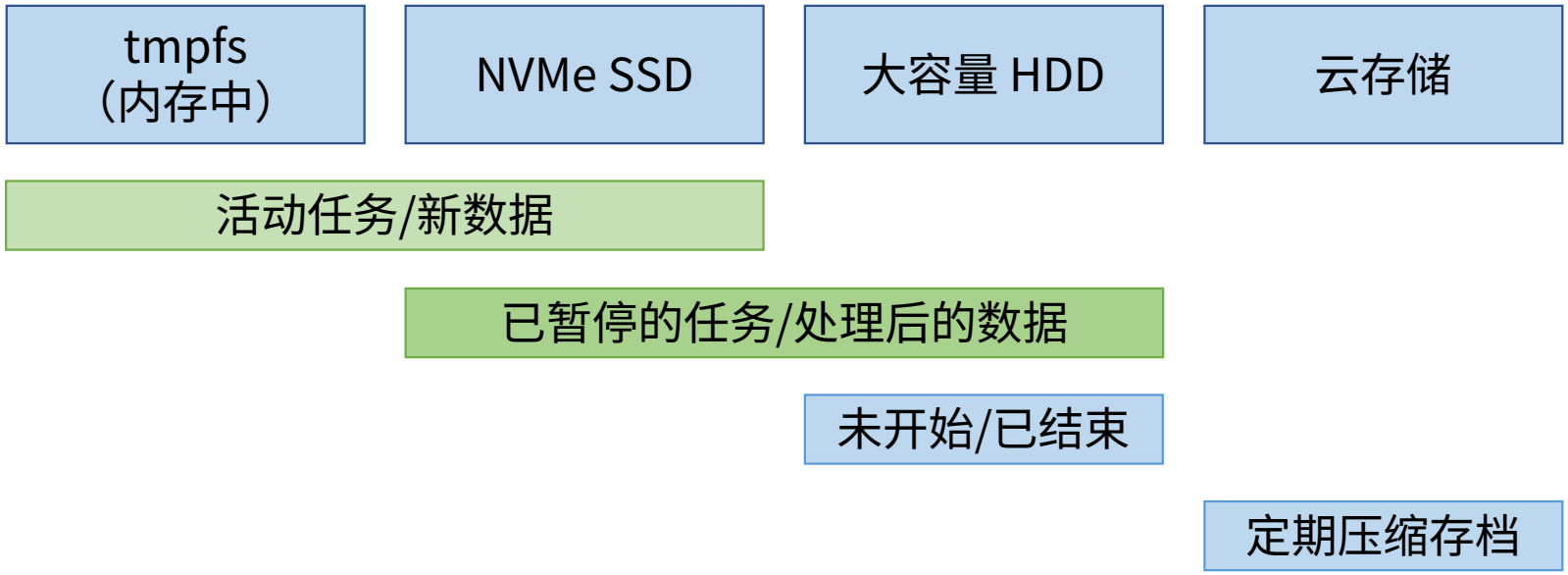
2. 系统构成

- **计算服务器**
  - 目前主要是 CPU 服务器
  - 均使用 **Debian 12**
  - 解决**中等核心数**、**高计算速度**的需求
  - **避免个人PC可能导致的问题**（电力供应、硬件管理等）
  - 繁重计算任务**不干扰其他办公工作**



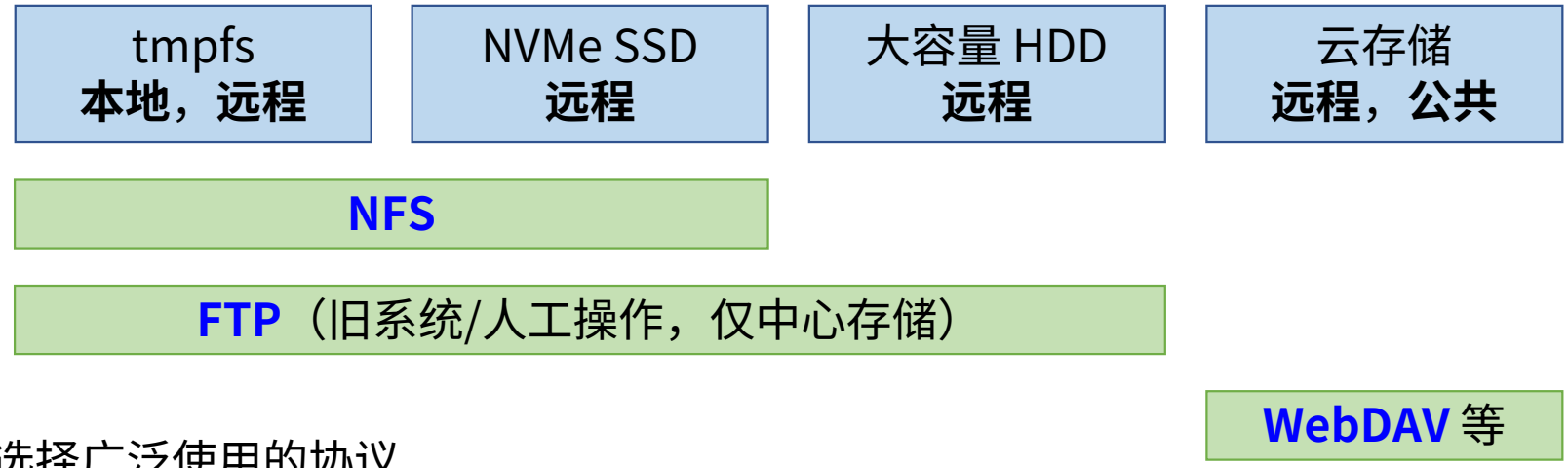
2. 系统构成

- **存储系统**
  - 按**速度**分层



2. 系统构成

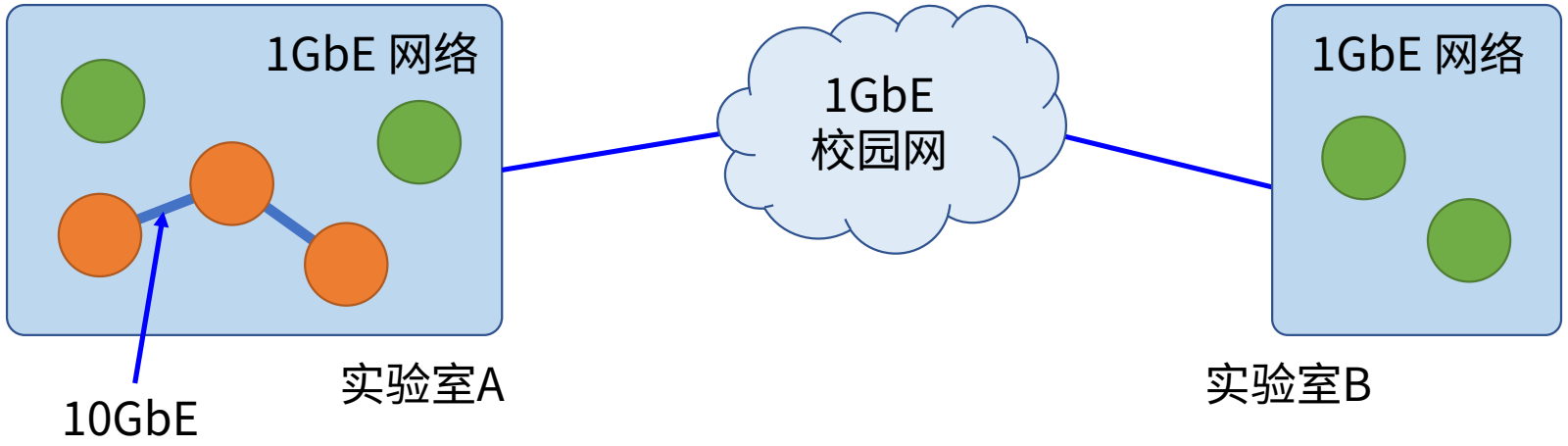
- **存储系统**
  - 本地&远程（**中心存储服务器**）访问



尽量选择广泛使用的协议

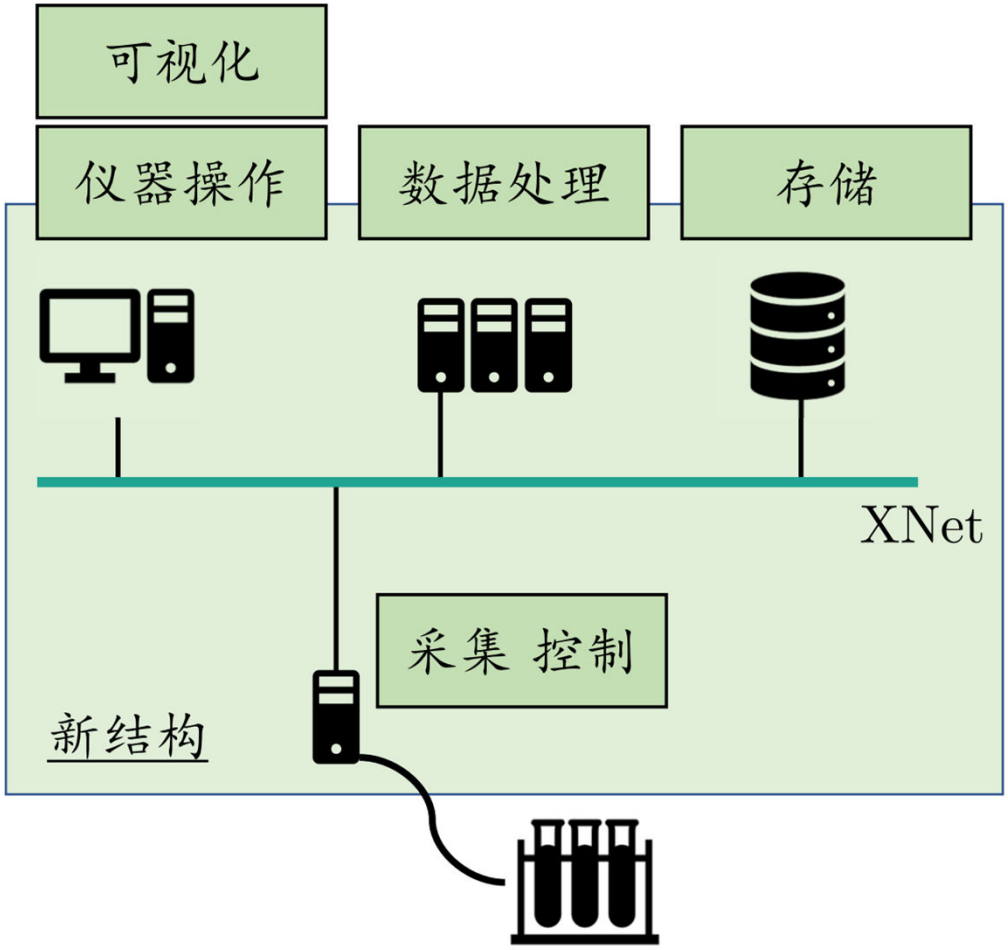
2. 系统构成

- **网络**
  - 基础：**1GbE** 校园网
  - 基础：**1GbE** 内部互联
  - 高速：**10GbE** 内部互联，**大数据量通道 RoCE**？专用 **RDMA 线路**?



2. 系统构成

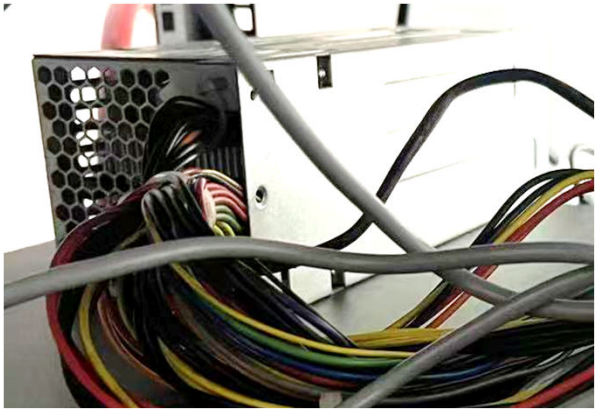
- **实验硬件**
  - 有完善的网络设施
  - 设备基本通过**有线网络**连接
  - 部分为**特定接口**（RS232等），加网关
  - 实验硬件使用 **XNet 协议**（*2024年开源软件论坛的报告*）



2. 系统构成

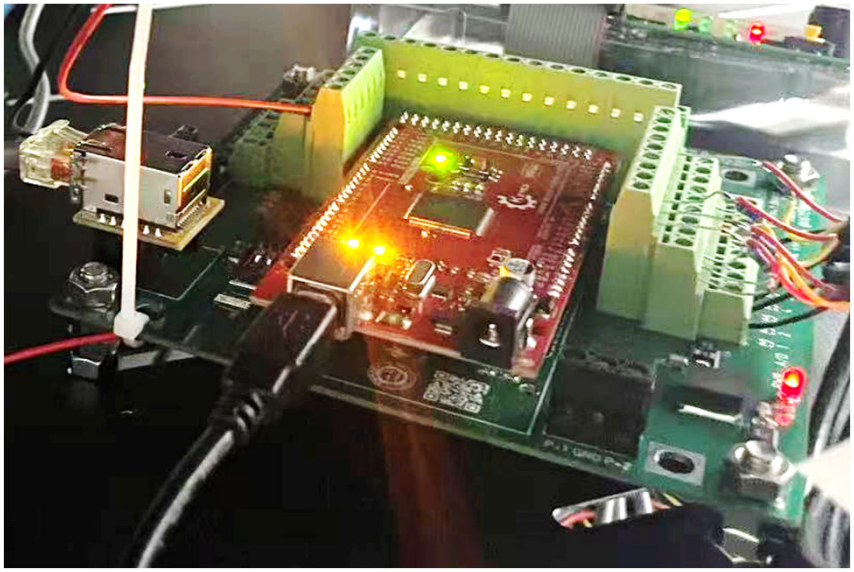
- **电源供应**
  - **220V AC（大部分服务器/网络设备）：UPS 电源系统**，确保电网掉电时能正常维持工作
  - **直流系统（板卡供电）**：双路自动切换的低压电源，可以用**冗余电源笼**

in



2. 系统构成

- **环境监测**
  - 自动化/无人值守系统必备！
  - 可以及早从监控数据中发现问题。例如，曾遇到空调故障，就是通过环境检测报警获知。
  - （多点）温度，湿度，VOC 等等
  - 自行设计的检测设备，同样使用 XNet



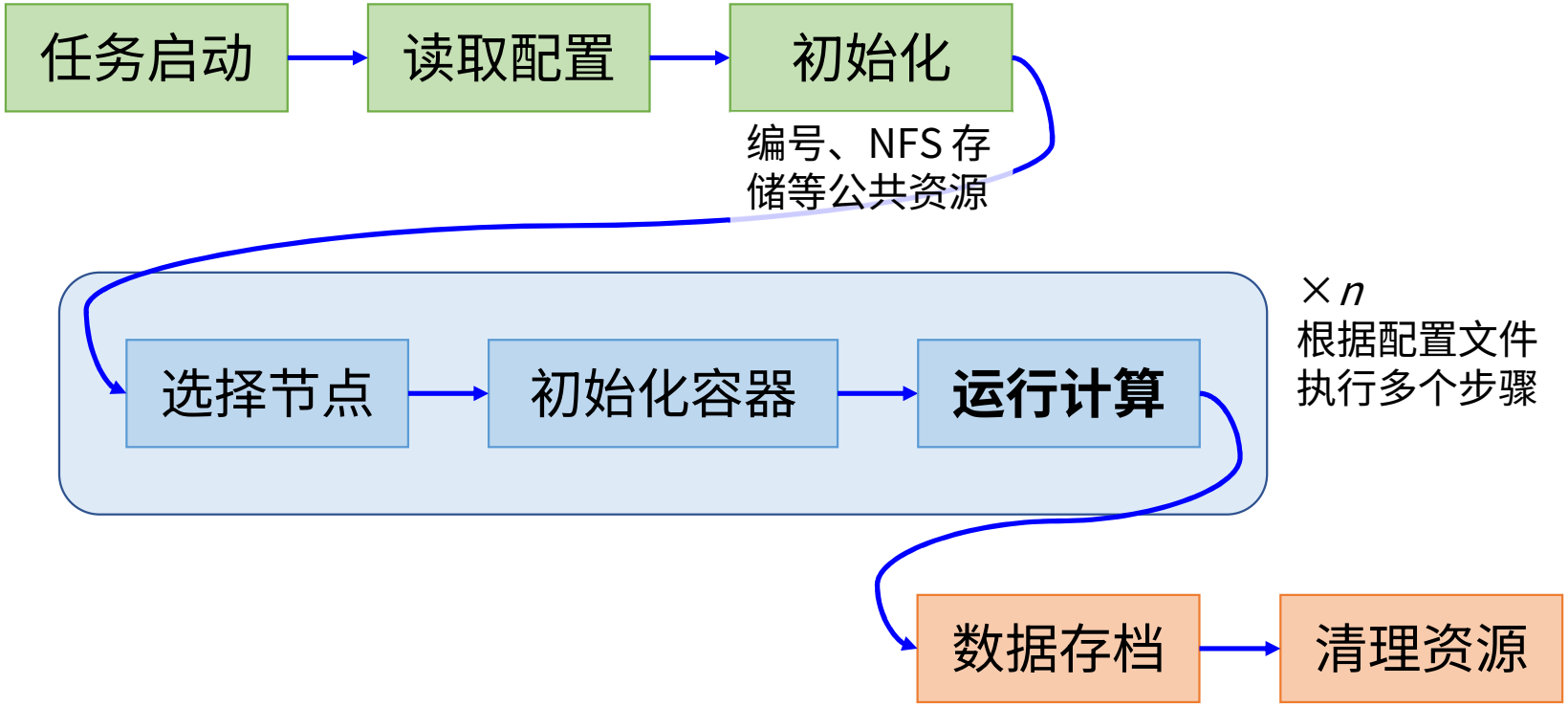
2. 系统构成

- **维护**
  - **关键任务上的日志&遥测**：任务产生的数据和运行状态都需要记录
  - **数据收集**：XNet(硬件) 或 Telegraf(服务器) – Prometheus – Grafana
  - **告警**：Grafana Alerting – 各类平台（邮件，企业微信等）
  - **备用或测试用的硬件**：硬盘，交换机，内存，温度探头，网线，等等

3. 业务

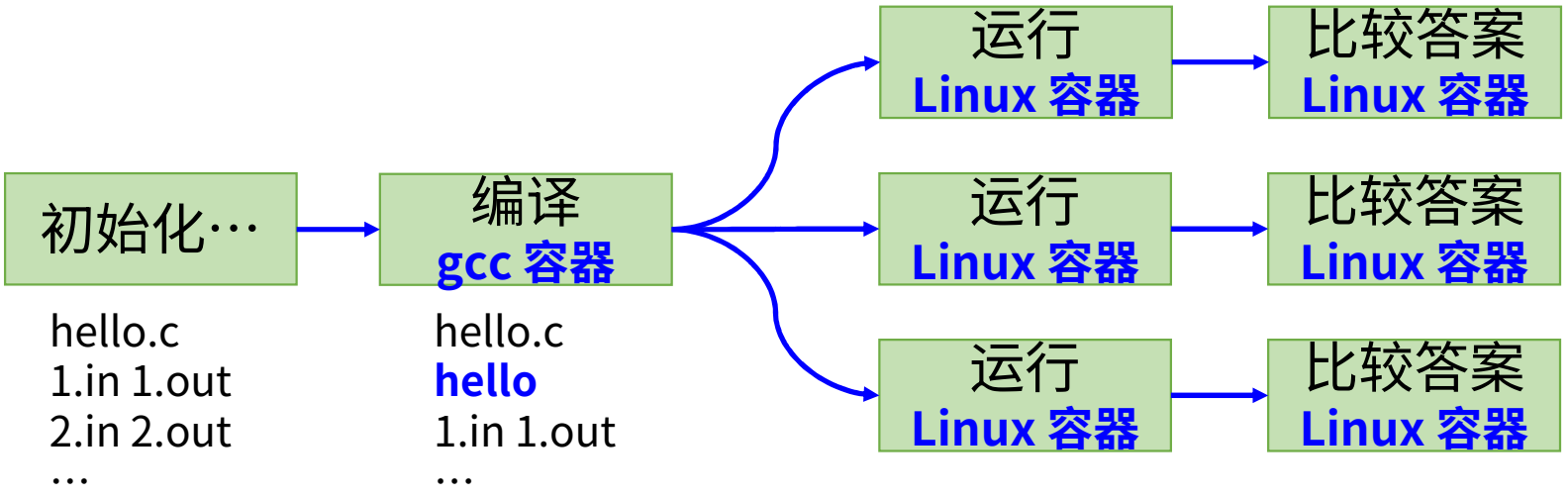
- 计算任务**统一化，流程规范**（scripting）
- 如何做到?
  - 借鉴 CI/CD，以脚本的方式**定义一个任务的过程**
  - 通过 Docker 容器或者 VM，**规范任务的运行环境**，确保可复现
  - 步骤间隔离，但有中心 NFS 进行**数据的集中存放和共享**，实现步骤之间的传递

3. 业务


任务启动
读取配置
初始化
编号、NFS 存储等公共资源
选择节点
初始化容器
运行计算
×n
根据配置文件执行多个步骤
数据存档
清理资源

3. 业务

- **例如：编译并测试一个 C 程序**，类似 OnlineJudge 的流程



3. 业务

- **例如：编译并测试一个 C 程序**，Taskfile 写法

```
compile-and-test:
    task compile param arch=amd64
    parallel (index, 10) {
        task run-program param id=index,arch=amd64
        task compare-output param id=index
    }
```

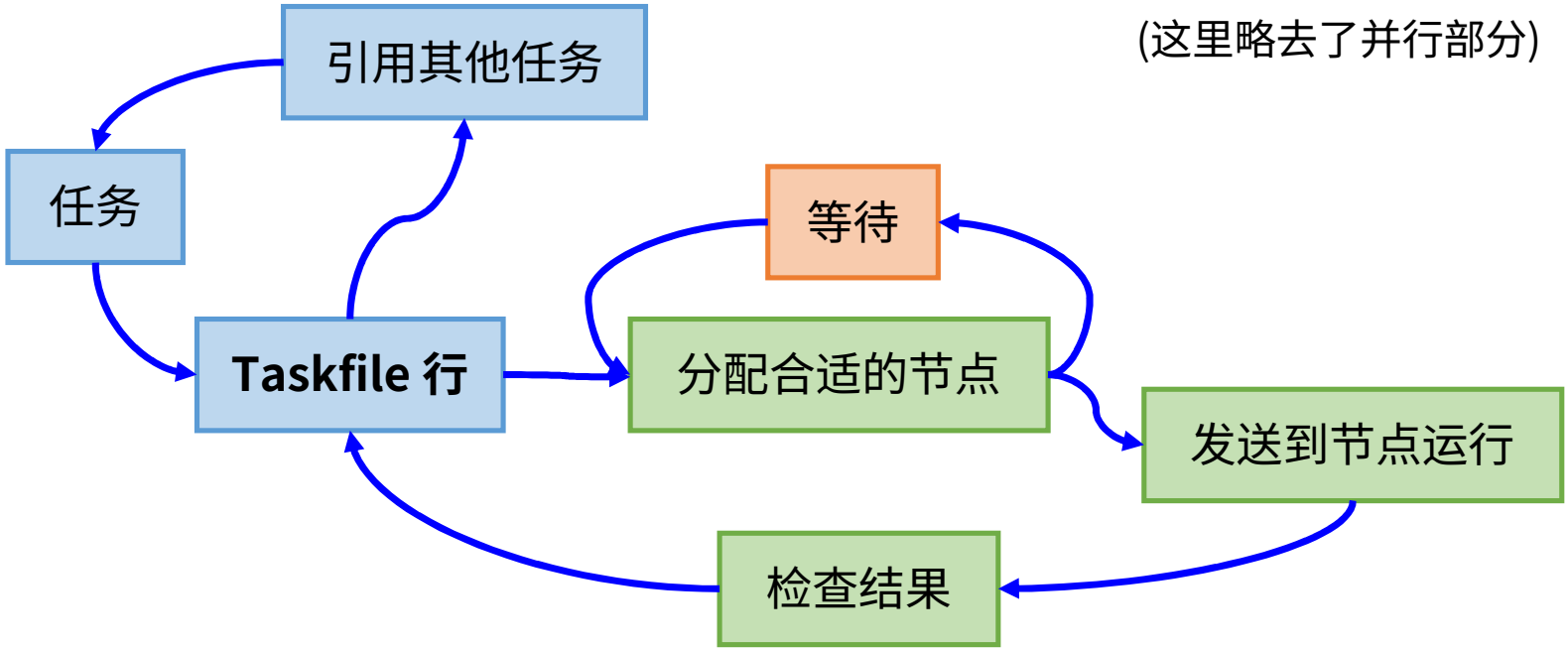
3. 业务

- **例如：编译并测试一个 C 程序**，Taskfile 写法

```
run-program: param id,arch
    require arch:{arch}    - 依赖 arch:amd64
    in linux               - 在 linux 容器运行
    run ./hello < hello{id}.in > hello{id}.out
```

- **Taskfile**：结合了 **Makefile** 和 **Dockerfile**
- **任务步骤**，**需要运行的命令**，**串/并行关系**，**依赖**等等

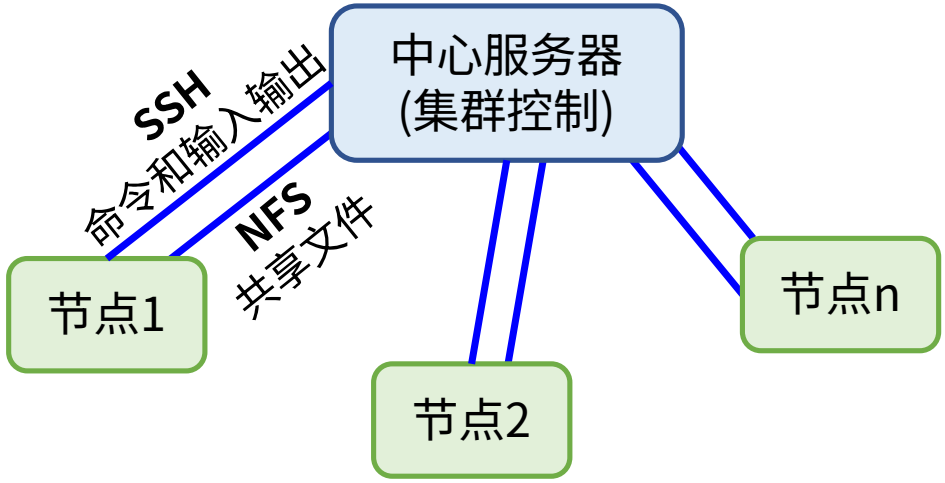
3. 业务



(这里略去了并行部分)

3. 业务

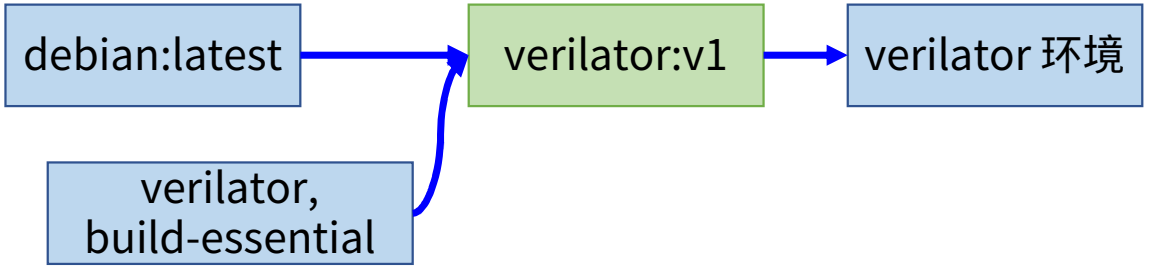
- 如何运行 Taskfile？中心服务器上运行。
- 最大程度利用**现有的协议（SSH，NFS等）**，同时进行任务调度。



任务从头到尾都在同一个目录（/work）下运行，这个目录的内容由NFS共享，所有各步骤的容器都能看到。
因此——不需要担心运行位置问题。

3. 业务

- **运行环境：Docker 打包**，或者在需要更好的隔离时用 VM
  - 有效避免了**环境构建的麻烦**
  - 放在中心服务器的 Docker Registry 上
  - 计算节点在需要时（Taskfile 中指出）自动拉取
  - 目前已经做过的：编译环境，常见的计算软件，Verilator等等
  - 容器环境定期更新，确保环境内程序的版本正确
  - 较独立的大型软件（/opt 那些）可以考虑做 readonly 挂载到容器



3. 业务

- **公共服务**
  - **为实验室内成员和计算设备提供服务**
  - **APT 镜像**：RISC-V 软件镜像源(Nginx)，为 RISC-V 节点提供软件更新
  - **内部网页**：各类通知，手册等(Apache2 + MkDocs，Gitee CI/CD)
  - **代码管理**：Gitea
  - **文件存储**：FileZilla FTP，仪器设备的手册资料/SDK/软件等等
  - **环境监控**：遍布实验室范围的传感器终端，Grafana 可视化
  - **OpenVPN**：内网访问，方便远程调试
  - 等等

3. 业务

- **系统管理**
  - 全面实现公钥认证
  - 多层级日志记录
  - 数据收集和可视化
  - 故障/异常告警
  - 访问控制

    物理层面；软件层面

4. 展望

- **新架构计算**
  - 在计算任务中更大范围地使用 RISC-V 等新型架构，利用好现有的 RISC-V 硬件设备(4*Lpi 4A, 4*Lpi 3A = 48 cores, 128GB DDR4)
- **更好的身份认证**
  - LDAP 等更适合集群的方案
- **GPU 计算节点**
  - 除了 CPU 计算节点外，可以使用 GPU 对部分计算进行加速
- **任务调度**
  - 精细化的任务调度策略，也许可以引入机器学习?

4. 展望

- **远程计算**
  - 原理上看，目前的集群设计已经可以支持远程计算，可能需要加网关
  - 充分利用本地边缘计算系统的优势，降低跨实验室的数据流量
  - 自动将大型计算任务 offload 到公共计算平台
- **共享集群**
  - 网络中类似结构的集群实现计算资源共享
  - 更加广泛且均衡地利用好计算资源
- **节点自动注册**
  - 节点运行 daemon，向中心服务器自动报告自身的配置状况，实现即插即用

5. 一些经验

- 避免大而全的结构：用一个协议做好一个事情
- 利用现有协议：SSH, NFS, FTP, HTTP, rsync, WebDAV
- 本地服务：特殊情况单独对待（RISC-V 本地镜像站等）
- 降低部署负担：目标主机（节点）上只需要有ssh就能部署
  - 更重要的是因为 RISC-V 架构不完整的支持
- Make it → Make it work → Make it right → Make it fast


Thanks