**项目总结报告**

日期：2025年1月13日

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 组号 | 5 | 项目名称 | 集群性能监控系统 | |
| 编程语言 | Shell | 开发平台和框架 | Prometheus | |
| **项目工作小结** | | | | |
| 1. 是否实现了项目立项时的所有需求？列出实现的新增需求和未实现的需求。   实现的需求：  1. 实时 GPU 健康状态监控：系统支持监控 GPU 的温度、功耗、显存占用率、核心利用率等关键性能指标，满足了实时性和准确性的需求。  2. 分布式 GPU 和节点资源监控：通过多节点的数据采集和分布式监控模块实现，适用于大规模集群环境。  3. 时序数据存储与可视化分析：使用 InfluxDB 和 Grafana 构建高效的时序数据库和直观的可视化仪表盘，实现了数据的长期存储与趋势分析。  4. 性能调优与资源优化：对 GPU 使用的动态调整、资源分配的优化提供了直观的参考依据。  新增需求：  1. 集群监控与容器化支持：新增对 Kubernetes 集群资源的监控能力，使系统能适应容器化和云原生架构。  2. Grafana 的增强功能：新增支持自定义仪表盘、动态查询以及复杂数据交互的能力，进一步提升了数据可视化的灵活性。  3. 用户操作日志追踪：为满足审计需求，新增了操作日志记录功能，提升了系统的安全性和可维护性。  **未实现的需求：**  • 自动告警与故障预警：基于 Prometheus 和自定义的告警规则，实现了对系统异常的自动告警和提前预警功能。  2.采用哪种架构风格？哪些设计模式？  **架构风格：**  1. 微服务架构：  • 系统由多个独立服务模块构成，包括 Prometheus（数据采集和告警）、InfluxDB（时序数据存储）、Grafana（数据可视化）和 DCGM（GPU 数据采集）。  • 各模块通过标准化接口进行通信，实现模块化和灵活部署。  2. 事件驱动架构：  • Prometheus 基于规则的动态触发告警机制属于事件驱动设计。  **设计模式：**  1. 观察者模式：用于自动告警，Prometheus 监控系统指标，并根据预设规则触发告警事件。  2. 分层架构：系统分为数据采集层（DCGM 和 Node Exporter）、存储层（InfluxDB 和 Prometheus）和展示层（Grafana）。  3. 工厂模式：用于封装不同硬件平台的监控模块，便于扩展新硬件支持。  4. 适配器模式：将不同数据源（如 Prometheus 和 DCGM）的采集结果标准化，统一传递给后续处理模块。  3.技术方案有哪些亮点？  专门化的 GPU 监控能力：  • 使用 NVIDIA DCGM 监控 GPU，获取详细的 GPU 性能指标。  高效的时序数据处理：  • 结合 Prometheus 和 InfluxDB，满足高频监控需求。  灵活的告警与扩展：  • 基于 Prometheus 的告警规则实现动态触发，灵活性强。  可视化与易用性：  • 使用 Grafana 实现实时监控数据的可视化，支持直观的仪表盘定制。  4.是否做了单元测试？是否做了系统功能测试？是否做了性能测试？是否做了兼容性等其他非功能测试？  5.是否采用大模型来辅助开发？对开发生产率约有百分之几的提升？  本项目未采用大模型（如 ChatGPT、Copilot）来辅助开发。 | | | | |
| **项目组成员对项目的贡献度（%）** | | | | |
| 吴伯涛25%  李双威25%  尚子森25%  申煜坤25% | | | | |
| **软件度量** | | | | |
| 软件代码行数（不包括注解行、空行和复用代码）： | | | | 3500 |
| 复用他人代码行数： | | | | 0 |

|  |
| --- |
| **经验、教训和建议** |
| **经验:**  选择合适的技术栈: 项目团队选择了成熟的开源工具，如Prometheus、DCGM、Node Exporter和InfluxDB，这些工具经过验证，拥有丰富的文档和社区支持，降低了开发风险。  面向对象开发方法: 采用UML进行系统建模，有助于团队成员理解项目架构和流程，提高开发效率和代码质量。  迭代开发: 将项目分解为多个迭代，每个迭代解决特定的风险，逐步完善系统功能，降低了项目失败的风险。  测试驱动的开发: 使用PyTest等测试工具，确保代码质量和功能稳定性。  **教训:**  需求风险: 项目初期需要充分与用户沟通，明确需求，避免后期需求变更带来的开发成本增加。  技术风险: 技术选型需要谨慎，避免选择过于复杂或难以驾驭的技术，导致项目延期或失败。  人力风险: 需要评估团队成员的技术水平和经验，避免因技术不熟练导致开发进度延误。  时间风险: 项目管理需要严格，制定合理的开发计划，并定期进行进度跟踪和调整。  安全风险: 需要重视系统安全性，采取必要的安全措施，防止数据泄露。  **建议:**  加强需求管理: 建立需求变更流程，确保需求变更得到有效控制。  进行技术预研: 在项目启动前，对选定的技术进行充分的调研和评估，确保其适用性和可行性。  提升团队能力: 为团队成员提供必要的技术培训，提高其技术水平和工作效率。  加强项目管理: 使用项目管理工具，如Jira或Trello，跟踪项目进度，并定期进行项目评估和调整。  重视安全设计: 在系统设计阶段就考虑安全性，并进行安全测试，确保系统安全可靠。  关注用户体验: 进行用户测试，收集用户反馈，并根据反馈进行系统改进。 |

项目组各成员签字：