附录 A: “QoS感知的并行远内存数据卸载优化”研究项目 (SOW)

目录

[1. 合作需求/任务 2](#_Toc353436880)

[1.1 研究领域 2](#_Toc353436881)

[1.2 研究目标 2](#_Toc353436882)

[1.3 交付成果 2](#_Toc353436883)

[2. 验收方法与指标 2](#_Toc353436884)

[2.1 验收指标： 2](#_Toc353436885)

[2.2 其他验收条件： 2](#_Toc353436886)

[3. 项目阶段 2](#_Toc353436887)

[4. 项目监控和管理 2](#_Toc353436888)

# 合作需求/任务

1.1 基本信息

项目名称：QoS感知的并行远内存数据卸载优化

合作方注册地址：xxxxx，邮编：

1.2定义（注：本条中应全面列举对与本合同相关的各专用术语的含义，按术语首字的拼音排序。本条的定义仅为明示有关缩略语定义写法以供参考，实际项目中请做相应修改）

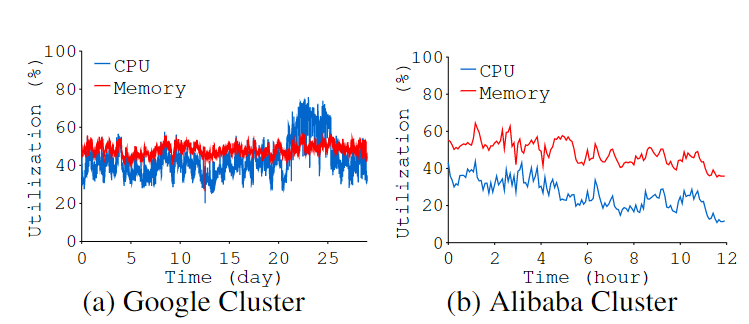
远内存（Far Memory），也称分离式内存（Disaggregated Memory），将计算资源如CPU等与内存资源解耦一种内存资源管理架构。在本项目中，远内存指代了用于卸载内存数据的所有可能的内存、存储硬件系统，包括但不限于：基于RDMA的远内存，基于智能网卡的远内存、基于持久内存（Persistent memory）的远内存、基于SSD的远内存等。

QoS（quality of service）指的是应用的服务质量，在本文中指的是应用的整体运行延时。

数据卸载指的是，我们将当前应用的一部分工作集（运行时数据）卸载到远内存中。

1.3研究背景

近几年，机器学习、键值存储等对内存具有较大需求的应用发展迅速，在数据中心的应用中占有越来越重要的比重。同时随着互联网时代的高速发展，不断膨胀的数据规模也对传统数据中心的硬件资源提出了挑战。据2020年一期《Science》上的研究显示，虽然数据中心能耗爆炸势头有所放缓，全球数据中心仍需技术创新以应对目前效率红利日趋殆尽的问题。但随着硬件设备的发展，摩尔定律逐渐走向尽头，内存设备的发展逐渐达到了瓶颈。DRAM设备制造商们也都在不断尝试获得更高的内存存储密度和更低的单位存储成本。从以上两点来看，内存资源逐渐成为了限制现有数据中心的发展瓶颈。

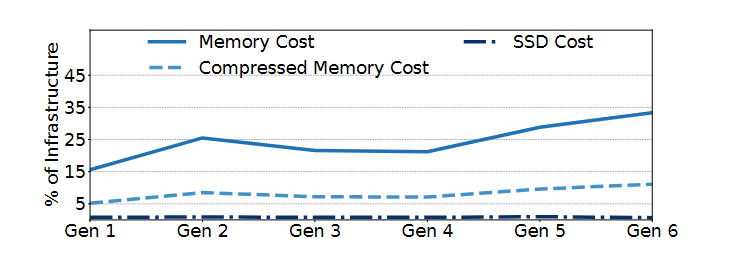


图表 1 数据中心资源利用率

而传统的数据中心架构一直以单片服务器作为部署和操作单元，随着硬件设备的发展已经应用场景的变化，传统的架构也面临着一些问题。

首先是内存资源利用率的问题。由于服务器是资源分配的物理边界，因此很难充分利用数据中心中的所有资源。就内存资源而言，由于应用对于内存资源的要求，需要足够的内存空间，这也就意味着通常会为单个服务器配置更高的内存容量。但由于高负载场景并不常见，这也就意味着数据中心大部分集群的内存资源利用率较低。在LegoOS这篇文章中作者调研了Google和阿里的生产集群[1]。图1展示了传统架构集群的资源利用率，对于这两个集群，只利用了大约一半的 CPU 和内存。这些生产集群中资源利用不足的主要原因之一就是作业的CPU和内存资源必须从同一台物理机器上分配，这也导致了CPU资源和内存资源发生倾斜的时候，存在资源利用率较低的情况。

其次便是内存资源的成本在数据中心中占比越来越大。在TMO中，Facebook对数据中心的部分资源的消耗占比进行了统计[2]。图2中展示了DRAM，SSD以及压缩内存在数据中心随着更新迭代的消耗占比走向。从图中可以看到DRAM设备作为传统数据中心不可获取的基本组件，其成本随着硬件的发展不增反降低。如果能够降低DRAM设备的支出占比，能够带来较大的成本收益。

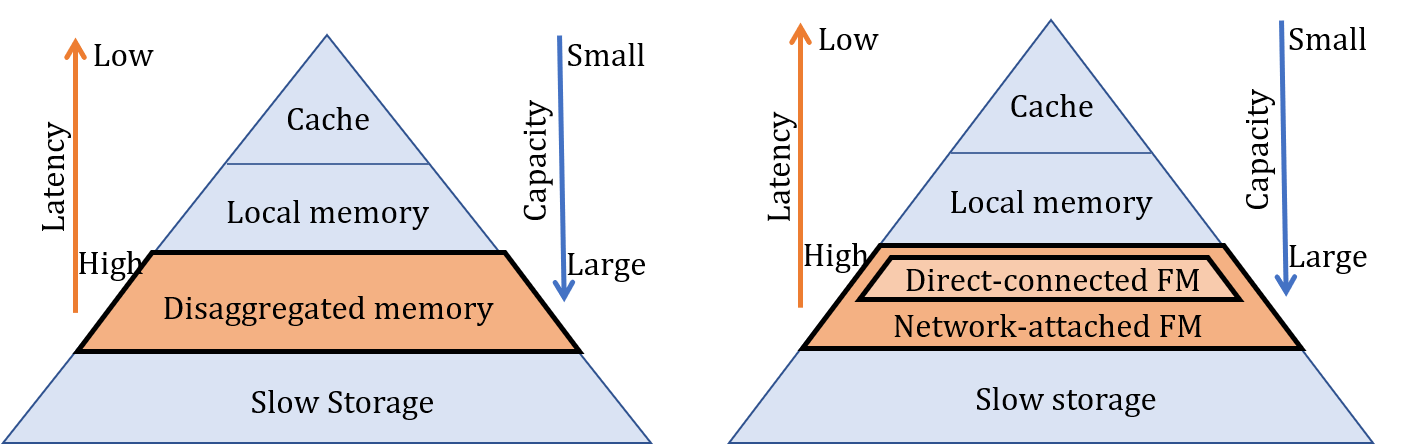


图表 2 内存、压缩内存和SSD的成本随硬件代占数据中心基础设施的百分比

近年来高性能、低成本的非DRAM存储设备诸如NVMe SSD已经出现，并已经逐渐部署到部分数据中心上去；新型的传输协议例如CXL，抑或是随着网络带宽增加性能大大提升的RDMA技术，为内存的连接提供了更为宽泛的选择。新型存储介质和传输介质的出现也为新的内存架构创造了可能性。如果能够使用更加便宜但性能不输于传统DRAM的存储介质能够大大降低数据中心的成本，而更加高速便捷的传输介质的出现也意味着内存的连接方式有着更加多样化的选择，内存也许不再需要和CPU资源高度耦合。

面临这样的问题以及技术背景，受到现有数据中心计算资源、网络资源、存储资源的池化的启发，许多人提出了分离式架构。内存资源作为计算资源的一部分，CPU与内存高度耦合，将内存资源解耦出来能够大大提升系统整体的资源利用率，提供更加灵活的内存服务。因此分离式内存的核心思想便是内存资源的解耦。

现有分离式内存架构虽然都是为了解决CPU与内存高度耦合的问题。但根据其实现层次和实现方式，最后的表现和特性也各不相同。图三是分离式内存在内存分级中的位置，可以看到分离式内存相比于传统内存结构有着更强的灵活性，实现方式也各有特色。首先是分离式内存的实现层次。由于内存的工作通路从应用层到系统层，最后再到硬件层，分离式内存的实现也不可避免地经历这些层次。在不同的层次实现也就决定了分离式内存架构的性质。实现层级更高意味着对顶层的透明性更差，也意味着对底层硬件设备的依赖降低，更容易适应现有集群。实现层级越低意味着对硬件设备有着一定的要求，但对顶层的透明度也更高。其次，分离式内存的一个重要因素便是内存的物理解耦程度。目前的分离式内存主要有两种实现，一种是将分离式内存作为本地内存的扩展，例如FastSwap[3]，另一种便是将内存和CPU完全解耦，直接进行远内存的存取，例如LegoOS, Cilo[1,4]等。前一种方式倚靠现有系统的swap，对上层应用有着良好的透明性，但由于仍然是通过原有内存通路，反而增加了远内存访问的消耗，不可避免地带来额外的开销。后一种方式直接进行内存存取，但难以和现有内存通路共存，一方是需要进行大量的逻辑设计，另一方面则是难以做到对应用层完全透明。最后便是从实际应用层面来讲，还需要考虑到系统的透明层级、系统部署成本、设备适用性以及系统的性能。



图表 3 分离式内存在内存分级结构中的位置

在现有分离式内存的工作背景下，本项目旨在研究一套能够实现应用透明感知的远内存后端进行并行卸载的系统。为了能够尽快将系统部署到实际生产环境，就必须实现较高的用户透明度。使得原有应用能够直接在新的集群上运行而不需要进行额外的适配修改。同时不同应用对于内存介质的敏感程度也有着不同的差异，在不同工作场景下负载情况也不同，如果能够根据应用的工作运行状态自适应调整内存分配结构，使得应用的内存合理规划，进而提高整体的吞吐率。最后便是混合内存后端的卸载。为了能够适应现有集群，依托原有内存系统进行多内存后端的扩展在便利性和性能上都有一定的优势。同时由于数据中心内存后端资源种类复杂，如果能对多种后端进行统一管理，根据应用的需求进行内存的分配，将会大大提升内存资源的利用率、降低内存资源的生产成本，提升应用的整体吞吐率。

总之，随着数据中心建设受到内存资源的限制，需要一套体系结构感知黑盒应用的工作状态，实现混合内存的自适应分配，提升数据中心整体的资源利用率，为实现内存资源的高度池化进行一定的探索。项目旨在为华为数据中心提供一套基于现有集群的面向多种QoS黑盒应用的并行远内存后端卸载优化系统。

参考文献：

1. Shan Y, Huang Y, Chen Y, et al. {LegoOS}: A Disseminated, Distributed {OS} for Hardware Resource Disaggregation[C]//13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18). 2018: 69-87.
2. Weiner J, Agarwal N, Schatzberg D, et al. TMO: transparent memory offloading in datacenters[C]//Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2022: 609-621.
3. Amaro E, Branner-Augmon C, Luo Z, et al. Can far memory improve job throughput?[C]//Proceedings of the Fifteenth European Conference on Computer Systems. 2020: 1-16.
4. Guo Z, Shan Y, Luo X, et al. Clio: A hardware-software co-designed disaggregated memory system[C]//Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2022: 417-433.
5. Wang J, Li C, Mei J, et al. HyFarM: Task Orchestration on Hybrid Far Memory for High Performance Per Bit[J].

1.4 研究内容及目标

本课题的目标是研究一套能够实现应用透明的混合远内存后端卸载的分离式内存架构。不同于现有手段，本课题将探索设计一套自适应控制黑盒应用的内存占用，以提高整体吞吐率。同时对传统架构进行改进，在现有系统上增加横向原内存和纵向远内存等多种内存卸载后端的支持。最后根据应用本身的内存使用特点，自适应进行横向远内存和纵向远内存的动态分配，实现应用的自适应高效透明运行和内存资源的充分利用。

1.4.1 拟解决的技术要点

本课题的主要内容是研究一套基于新型内存架构的黑盒应用内存资源分配方法，使之能够在各种QoS场景下根据负载情况进行内存资源的合理分配，能够根据应用本身特点实现差异化的内存资源限制，提高整体吞吐率，实现内存资源的高效利用，并为内存池化进行一定的探索。

**本项目中拟解决的技术问题具体包括以下两点。**

**Part1：黑盒场景下任务内存资源自适应调度**

1. **任务对于不同内存后端的敏感性分析**

我们将通过对任务各种页面的实时监控，分析任务对于不同后端的敏感程度，评估不同内存后端对于进程性能的影响。最终设计自适应算法，根据进程匿名页、文件页等页面资源的持有情况评估进程需求，以实现随任务运行状态变化进行分配的适时调整。

**B．任务对于内存的实时需求分析**

在TMO这篇文章中Facebook引入了PSI作为进程的资源压力的评估指标，然后根据该指标调整内存的卸载容量。受此启发，为了评估应用在运行过程中对于内存资源的需求，我们同样会引入进程对内存的需求指标，在此基础上设计自适应算法，实时对应用的运行状态进行监控，保证各个应用的正常运行。

**Part2：多后端内存卸载架构的设计与实现**

1. **现有内存回收机制的优化**

为了适应应用层的基于黑盒应用的自适应感知内存回收控制，需要对现有内存回收机制进行一定的优化。现有工作环境使用的内存回收机制普遍使用LRU算法进行内存的统一回收，没有区分不同页面之间差异化卸载。同样由于需要支持并行应用的内存卸载，也需要对内存机制进行一定更改，加入不同后端的控制逻辑和实现逻辑。

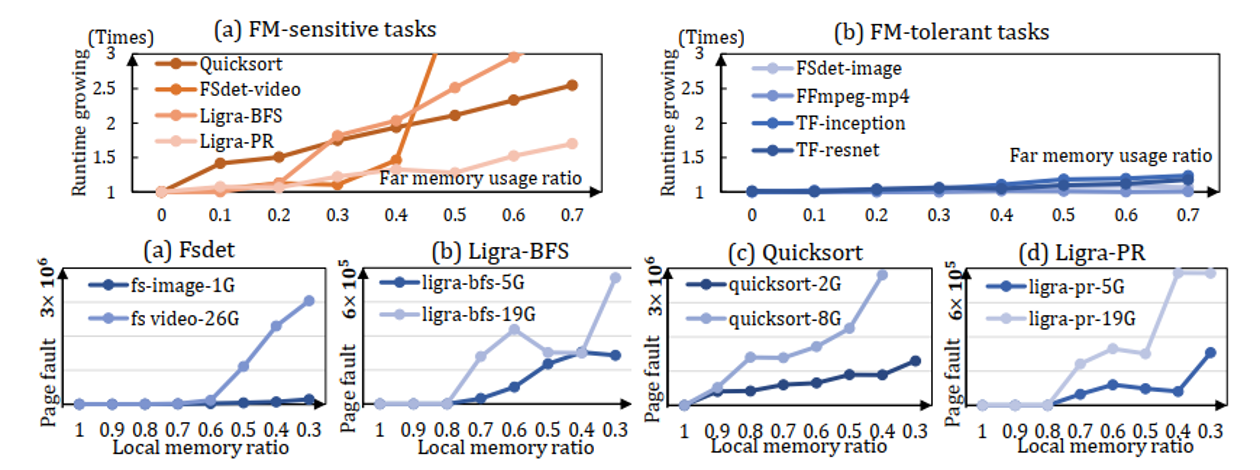
1. **并行远内存数据卸载架构的设计**

当前远内存数据卸载主要依赖swap机制，而当前为了支持并发运行的多路应用，他们需要通过同一个基于swap的内存卸载通路，而当前的swap机制仅支持串行的页面卸载，运行效率较差，并且由于多路数据卸载量增大，势必会带来swap的竞争，甚至整体延时还会提升。为了提高多任务场景下内存卸载的性能，实现远内存的并行卸载能够提升整体效率，进而提升任务的吞吐率。

1.4.2 研究方法

1）任务对于不同内存后端的敏感性分析

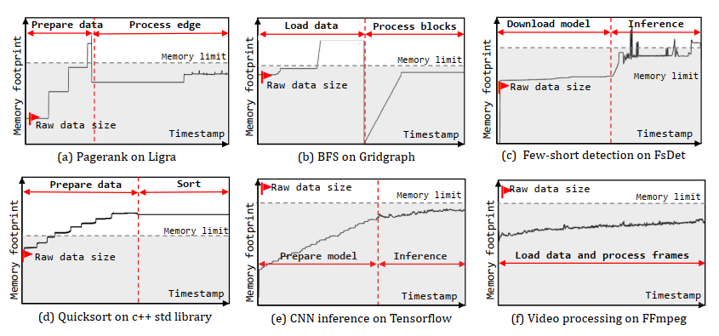
从图4中可以看到，不同的应用在远内存环境下拥有不同程度的性能影响，在远内存环境下的内存资源分配还需要考虑任务对远内存的敏感性[5]。但由于现有工作大多面向白盒应用，在已知应用的条件下进行不同远内存后端的调度优化。由于应用的复杂情形，导致这种模式难以应用到实际生产环境。因此我们会设计一套自适应算法，根据进程的实时运行情况，包括page fault数量，进程持有不同页面类型数量等，分析进程的内存特征，选择最适合该内存的内存卸载后端，进而提高进程的运行表现。



图表 4 不同应用在不同远内存和本地内存应用场景的运行表现

2）任务对于内存的实时需求分析

从图5中可以观察得到，应用的远内存敏感性随着任务进行会发生变化，任务的内存资源分配需要充分考虑动态变化，灵活回收和分配内存资源[5]。现代操作系统例如Linux使用的Cgroup提供了对内存实现各种限制的方法，使得能够对内存的资源分配进行灵活调整。而对应用的运行情况进行实时监控，评估内存的当前对于内存资源的需求，合理进行内存资源的回收，不仅为单个应用的运行提供保证，而且能够提升整体的资源利用率。我们预计调研对进程对于内存资源的需求指标，在此基础上设计自适应算法，实时调整各进程的内存资源分配。



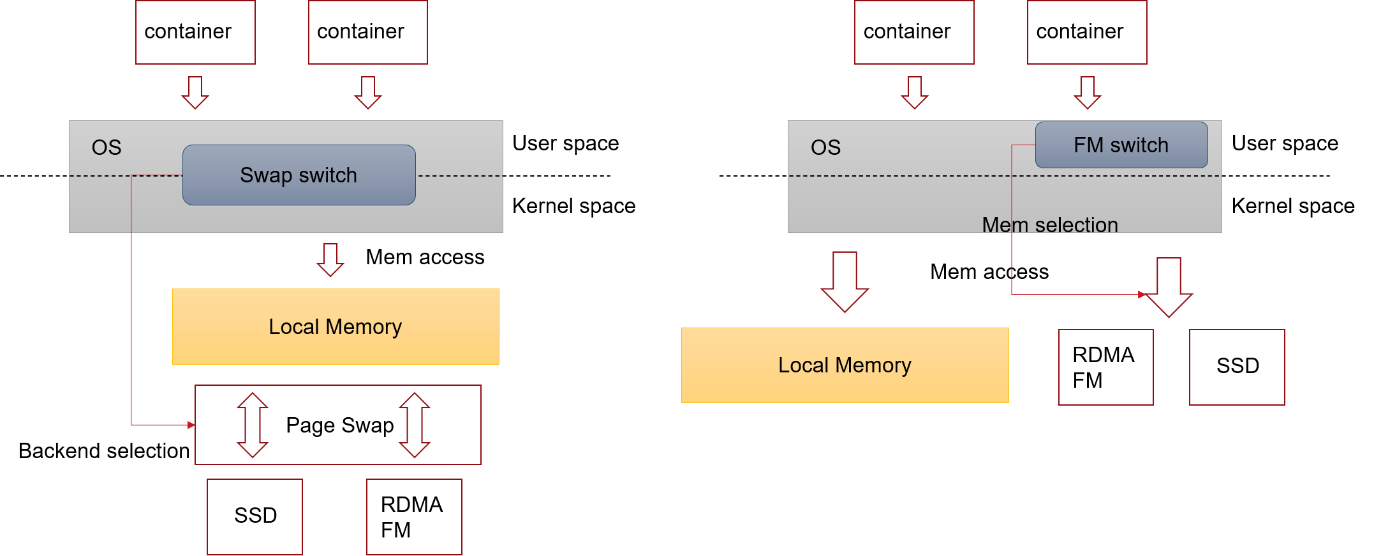
图表 5 不同应用随运行时间的运行状态变化

3）现有内存回收机制的优化

为了适应现有上层的内存控制逻辑和底层的远内存架构，我们将对现有内存回收机制和相关算法进行优化。由于传统操作系统设计到了内存回收层次是以页面为单位进行内存的存取，因此内存回收的有关机制会忽略到不同进程之间的差异性。首先我们会对现有内存回收机制进行调研，并结合一定的预实验，分析其中的设计没有考虑到进程差异化的地方。最后结合上层调度器和底层内存卸载架构做出一定的优化修改。

4）并行内存后端卸载架构的设计

我们经过前期的调研，对于多后端的远内存架构已经有了一定的设想。如图六所示，该图展示了我们目前对于远内存后端的的设计，这两种设计也代表着现有远内存的两种主流设计：一种是基于原有swap机制，将远内存作为本地内存扩展的远内存架构；另一种便是远内存和本地内存处于同一并行层级的远内存架构。分析可以发现，基于swap的架构能够较好的适应现有内存系统，能够依托原有系统的设计进行实现，但这类架构仍然依靠Page fault完成内存卸载。Page Fault作为内存的扩展机制，本身需要耗费较多时间，如果在此基础上再加入混合后端的选择，将会带来额外的开销。但如果能够在swap的基础上进行优化，使得各种后端的swap能够并行存在，预计会带来性能的提升。第二种架构把远内存看作独立于本地内存的层次，使得应用能够直接进行内存的存取。这类架构直接进行内存存取，无需经过原有内存的数据通路，性能较好。但此类架构同样存在一定的问题，在实现的时候预计需要应用层进行内存存取逻辑的实现，透明程度较差。我们将会在第一种架构的基础上进行研究，预计初步实现单个共享内存池后端，结合应用层的内存资源调度和系统层的内存回收构建完整的架构。之后在此基础上探索并行后端卸载架构的实现。



图表 6 两种远内存架构设计示意图（左图：基于Page Swap的远内存架构；右图：并行于本地内存的远内存架构）

1.5 交付成果

1. 软件交付

关于“基于QoS的任务内存资源自适应调度”以及“并行远内存卸载架构的设计与实现”两个技术的原型系统代码1套

1. 文档交付

|  |  |
| --- | --- |
| **序号** | **交付** |
| 1 | 基于QoS的任务内存资源自适应调度技术报告 |
| 2 | 并行远内存卸载架构的设计与实现技术报告 |
| 3 | QoS感知的并行远内存数据卸载优化相关专利2篇 |
| 4 | QoS感知的并行远内存数据卸载优化相关学术论文1篇 |
| 5 |  |

# 验收方法与指标

2.1 验收指标：

2.2 其他验收条件：

# 项目阶段

*（红色字体可以根据项目情况更改）*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **项目阶段** | **一** | **二** | **三** | **四** | **五** |
| **起止日期** | 2022.11.1~2022.11.31 | 2022.12.1~2023.2.28 | 2022.3.1~2022.5.31 | 2022.6.1~2022.8.31 | 2022.9.1~2022.110.31 |
| **工作描述** | 调研多后端架构实现方案；调研应用内存资源自适应调度的实现方案 | 设计与实现多后端架构，在实际集群上部署并行远内存架构 | 设计与实现应用内存资源调度器，实现内存资源的后端调度和内存容量调度 | 结合两部分技术，在实际集群上进行实验，优化性能 | 验证实验场景，并结合实际生产环境进行优化 |
| **交付件** | 技术报告 | Demo、技术报告 | Demo、技术报告 | 论文、原型系统、项目报告书等 | 源代码、项目报告书、专利、论文等 |
| **验收标准** |  |  |  |  |  |
| **验收方法** |  |  |  |  |  |

T：合同生效日期

# 项目监控和管理

本项目采用如下项目管理机制：

1. 项目双月月报。
2. 每隔6个月的面谈交流项目进展。
3. 根据实际需要安排的其它交流，如电话、邮件、电话会议等。

# 项目投入

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 姓名 | 职务/职称 | 在开发工作中的职责 | 投入的工作量  （单位：人•月） |
| 1 | 王靖 | 博士生 | 研发人员 |  |
| 2 | 杨涵章 | 硕士生 | 研发人员 |  |
| 3 | 刘一博 | 硕士生 | 研发人员 |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |

项目负责人相关信息：

|  |  |
| --- | --- |
|  | 项目负责人 |
| 姓名 |  |
| 职务 |  |
| 地址 |  |
| 邮编 |  |
| 电话 |  |
| 电邮 |  |