问题

项目general问题

1.1 什么是硬实时，是什么场景，什么时候要用到硬实时？

什么是？

硬实时保证**任务**在各种情况下一定能在**截止时间之前执行完**（否则就造成重大地安全事故）。比如说现在的自动驾驶或者航空航天等领域对硬实时有很高的要求。

为什么做？

硬实时保障，时间保障是安全关键系统正常运行的基础。

【场景】

比如说自动驾驶，工业控制，智能制造，工业机器人等等场景。

1.2 为什么做考虑互斥同步的调度/为什么着重强调互斥同步？【需求】

因为现在系统设计得越来越复杂，系统任务之间会存在频繁的协同和通信，而这都是由互斥同步来实现的，所以现在的系统存在大量互斥同步访问的资源，比如说一块内存块、一段代码片段或者一个IO端口，但是由于对资源是互斥访问的，需要锁。所以多个任务，尤其是多个并行任务对同一资源的访问就导致非常长的资源争用和任务阻塞，会极大地延后任务的完成时间，导致整个实时系统的不可调度和故障。

互斥同步的时延是实时系统的性能瓶颈。

1.3 贡献、特性、创新性：

两大贡献：

（1）通过我们提出的新的调度策略，为新兴复杂实时系统**提供硬实时（硬性时间）保障**，(可以应用到工业控制等领域。)

（2）同时也为linuxRT提供一个多核考虑共享资源的调度方案。

工程贡献：

我们把现有的理论知识MSRP完整实现/支持在了linux上（不是理论创新点）

创新性：

考虑阻塞的任务分配机制RAF、面向资源的优先级排序算法SPO：资源感知型 考虑资源争用导致的任务阻塞

1.4 这个idea是如何想到的？/ 这个调度是你们做的？

idea是老师的，我们主要是配合/协助老师一起做，同时把它完整实现在linux系统上（加了RT Patch补丁）

1.5 怎么保障实时性？如何验证

响应时间分析验证的（可调度性实验）。这个响应时间分析是基于我们的调度和分配，还有资源共享协议做出来的

给定一个系统，我们首先要拿这个响应时间分析去分析它。在我们的这个调度器下，我们这个调度器是配合着一个时间分析的，如果时间分析通过了算出最坏情况，我们能够给出这套系统在最坏情况下是否可调度。如果这个时间分析通过了（响应时间建模部分**算出了最坏响应时间**），比较其他算法发现它们在最坏情况下无法保证，那么我们就可以**保障这套系统在任何情况下**，它**每一个任务都可以在要求的时间内完成**。这就是**硬实时保障**。

（单元测试时间调度也是算出来的，最后的真机测试 是真实情况跑出来的）

1.6 什么是可调度性

系统中每一个任务的响应时间不超过ddl

能完成就是可调度，反之就是不可调度

也就是说只要这个系统里面有任何一个任务miss deadline，或者是运行超时了，那么整个系统就不可调度了，

这就是**硬实时系统的可调度性**

1.7 第二点目标要和CFS调度器做性能比较，为什么没有？

是这样的，因为我们做的是面向硬实时的系统，而CFS，它的算法的分配逻辑是根据每个任务执行的时间的比率分配它的执行时间，首先它就不是硬实时的一个系统，比如说虽然说他的执行时间很短，但是他需要的优先级很高，它是非常紧急的任务，那CFS它就已经没办法达到这样的效果。因此我们还是认为我们每个算法都应该具有一个优先级，我们还是跟一些实时系统的一些优先级的一些调度算法进行比较。我们对比的，现在操作系统里面很主流的调度和分配算法，比如说。DMPO或者Any-fit，这些都是主流的一些调度和分配的算法，我们明显比他们有性能上的优势。

1.8 真机结果 你们的方法前4个线程有两个ddl很后面

可以说是因为deadline短了，时间变的紧迫了，执行的很频繁，提升了系统的利用率。叫提升了系统的利用率就是让系统在一定的时间做更多的事。我们把它利用的更好了，它利用率上去了，我们的吞吐量才上去了

（就不要跟利用率挂上钩（现场很少专家懂硬实时系统，对他们来说，指的是资源利用率不是任务利用率）不要说降低利用率）

1.9 基础知识

（1）什么是**共享资源**：需要被**多个线程**访问的资源。

（2）**资源争用**：

是什么?

通过任务访问资源的频率，估计它们之间会产生的阻塞。

怎么做?

将争用资源最激烈的任务放在同一个核心，降低核间的争用，

为什么要降低核间的争用?

是因为核间争用会导致其中有一个任务，他在做无谓的自旋浪费CPU的cycle时钟周期，会极大程度上降低了系统的利用率。

1.10 开发方面的技术细节

讲的时候，先不说是litmus，可以说是基于linux和RT Patch补丁（提供实时调度的补丁支持）等实现的

等被问的时候，再说参考开源实现litmus（估计评委都没听说），插装yat调度类，以及其他一些修改（MSRP...），然后转移话题说我们的项目侧重点在于调度策略/算法上的实现（SPO、RAF）

1.11 为什么选择MSRP

（1）准确约束WCRT （不是wcet，说这是笔误，因为wcet是能测出来的），不可抢占，准确的时间保障

（2）MSRP保障访问资源任务的时间，两个算法同时保障访问资源和不访问资源的时间

1.12 如何解析程序参数

（1）**WCET**：LLVMTA分析工具和实测 两种途径结合得到

（2）程序里有哪些**共享资源**，以及**被访问次数**：人工分析（老实交代），future work，未来会提供能够自动分析程序中共享资源都有哪些，他们都被谁访问了多少次，被哪个线程访问了多少次等等（画大饼）

1.13 为什么不采用其他论文的非实时（比如陈渝省电模式）方案

其他方案也很有意思，但目前实时系统因为它要保障实时性，它就是要保障这个系统的最坏响应时间可分析。我们没有考虑更多更复杂的系统设计方案了，因为它会极大程度上增加我们对系统验证的挑战和难度。

【上面是general的，下面是演讲修改细节】

细节问题

1. 算法谁提的？你们的主要工作是？

老师提的

两个方面，第一个方面是协助。配合协助老师一起完成算法的设计。同时，我们把它完整地实现在了linux上。给linux提供了一个完整的能够考虑资源阻塞的任务分配机制 / 优先级。

（强调我们工程性上Significance。重要性）

2. RAF为啥要design成这样子。这样子能给我们什么好处？

争用资源最激烈的任务放在同一个核心，降低核间的争用。

为什么要降低核间争用？是因为核间争用会导致其中有一个任务，他在做无谓的自旋浪费CPU的cycle时钟周期，极大程度上降低了系统的利用率

3. RAF如何评估资源争用？

（通过**任务访问资源的频率**，估计每个任务之间的资源争用）

~~FIFO自旋锁：通过任务在一段时间内对资源的请求数量来估计每个任务之间的资源争用~~

4. 什么叫需要特定的可调度性测试 （悲观

现有的一些基于搜索的方法，他们需要去测试任务的可调度性