刘晓义(00:00:54): 配得上去吗？我只用 OBS 肯定是能推上去的，但是我直接用 OBS 的话，大家可能用我电脑讲会比较方便。是因为腾讯会议没有办法直接推直接用你电脑讲行，你这样讲都可以。但是有远程的同学。大家可以稍等我五分钟，我要重新定义一下。看看我能不能把 R TMT P 自己给配置出来。

刘晓义(00:02:00): 确定这是你把我给写好了就好。我不确定。我当时就是这样的，你当时是这样意思的，我当时这样就行。我要现场配置一下。你你当时是哪个不要慌。34还是10点这么高级卖网?

刘晓义(00:02:39): 一点听不懂听课你有可能是？空了，为什么是密码登录，为什么是怎么回事背诵？首先配合一下吧！1935后边儿忘了点儿东西。

刘晓义(00:03:33): 主要是这块儿要写啥，我给忘了。我又没有网，就是很僵硬。油。我有 chunk size 只需要写一个 chunk size。chunksize40为什么它会默认选一个4000一个数字？我也喜个来吧！没有爱情游戏就是没有中午快装了这个模块难道要重启它也不用了。rtmp server listen 我都不知道 R tmt server listen。Http R tmt nt. Thanks for reading. 我重启一下。是不是他杠大，我觉得你说的有道理。不是。这是什么？没有 RT MV，我装了这个模块。Live那个 usr live。Njx modules. 有我漏了他一下。怎么漏的来？Injects load module. 能不能在15分之前搞定15分之前搞不定就不能再重新定义了，就是 load module。是这么写的吗？modules. Etetc. 有买多是这样的，奥莱斯放屁。Open access. What th e**. 那是肯定了，你不要思路吗？发到 access log 里面。点 P 点2.3T。unknown log format. 好不清楚。Ok 然后见证奇迹的时候到了。我家自带了一个出墙的东西。Live.

刘晓义(00:07:15): Thousand of n. Life 这是他什么的 live stream 我能不能推流？看看。

刘晓义(00:07:52): Push your L third party. 我自己。

刘晓义(00:08:19): 单踢不走，铭文打出来了，乐什么没问题的 TM PK。enable. 来看看有没有戏。那房子就是。通过直播软件将你的视频发送给我们，即可开始直播，看来没有什么戏。我就是我老哥。这是 access log。空的 journals。看到这也没什么问题。老老的那还有个活着吗？现在的活都没戏了，配不起来了。我用 OBS 推了这个不好意思。激情配置加 inject 最后失败 settings。Stream primary. Connect. 选择你的品牌或账号？只能用用那啥腿不要用这个。Display culture 为什么默认是 display culture screen C 默认确实就是 aomatic culture. 不必要。Star t swimming. Hi people. 关联直播软件已开始直播流状态良好类型科学与技术。

刘晓义(00:10:19): hello everybody。这个重新定义了。由于腾讯会议还是没有找到很好的办法推 youtube 的流。那就只能这样了，我这两个流。为什么非要为什么是因为昌老师毕业了，我们之前白嫖的 zoom 没有。是不是很妙，是不是很符合出纳的这种感觉？Anyway. 不过密的 google meet 的话，主要是担心讲者不好推流就是不好，大家不好参与。主要还是因为有什么，你们的 google meet 是在浏览器里整的浏览器里的开会体验怎么样？前天咱们两个已经体验过。

刘晓义(00:11:22): anyway 这个不多说了，线上的朋友们你们好线上有什么问题吗？重新定义。七点钟的环节已经正式结束了，不好意思。好，可是等的时间比较长。你看。出纳技术群有没有人说发发生了什么问题？定义环节结束了？那就是今天咱们的 tonight 就正式开始，然后我就首先。今天。刚好上课，那时间很正好，对我甚至可以插上摄像头在这里 where is it. Visits.

刘晓义(00:12:52): 为喂，对我现在可以用麦克风了，我不仅可以用麦克风，我还可以 start video. 大家可以看到。我看不到我自己。腾讯会议室大家看怎么用，anyway 就是今天咱们的 tonight 可以正式开始，然后今天是重大的。可以庆祝的日子是因为 OS UP 已经正式结束，然后今年咱们的四个项目又是完全成功。然后这个完全通过都说明明年我们也可以有四个项目。而不是只有三个项目或者两个项目，甚至只有一个项目。咱们今天好给大家鼓掌!

刘晓义(00:13:49): 然后咱们今天邀请到的是包括24年和去年，因为去年并没有办这个结项的分享的环境，然后邀请了去年的同学和今年的同学一起去分享一下自己这个项目里完成的一些工作，然后一些心路历程。主要是希望大家听一听大家的心路历程，开发过程中遭受了怎样的痛苦磨难。

刘晓义(00:14:36): 孔宇飞同学在线上吗?

Conqueror712(00:14:40): 喂。hello. 可以听到吗?

刘晓义(00:14:46): 看到你了。我把 cos 给你。那成就是孔玉飞同学今天是没有来到现场。

Conqueror712(00:15:09): 喂。可以听到吗？我看会议里面是显示话的，我不知道是不是能听到?

刘晓义(00:15:25): 稍等一下，我这我们这边的外放好像有点问题。

刘晓义(00:15:40): 可以再说一下话吗?

刘晓义(00:15:45): Ok ok 没什么问题了，但是我们这边现在暂时没有看到共享屏幕，是不是没有看到共享屏幕?

Conqueror712(00:15:46): Ok. 我现在共享屏幕。我来共享一下。

刘晓义(00:15:53): 别看到。

Conqueror712(00:15:57): 可以看到吗？是可以看到的。ok 那我就简单的来说一下我是这个2023就是去年的参加项目的成员?

刘晓义(00:16:04): 没什么问题。

Conqueror712(00:16:16): 我今天简单来分享一下我主要是分享一下心路历程，就是像刚才说的。然后具体的细节因为我也。隔了一年了，有点忘了。我就简单讲一下。然后因为等一会我还有一个别的会，所以我就快速的过一下，那我当时是实现了一个做的是用户态的模拟器，那我们就来看它是怎么回事。首先，我们来介绍一下。不用多说哭了，那 nvidia 这个东西。现在是非常好，有很多的。非常良好的特点，像是并行计算。内存层次结构优化等等。那我相信没有一个人没有用过。然后但是我们知道它是一个必然的东西，或者说它绝大部分的东西是一个必然的，在。

Conqueror712(00:17:22): 用 C 来打包一些，就是你来 C 来写一些东西，然后在打包的时候会有一些各种各样的神奇的问题就是包括没法就很难，这个重分发或者是一些进行测试，也是比较跟开源的相比肯定是比较困难的，那这个时候我们需要去花一些额外的。人力物力去操作，那我们然后包括要依赖一些可能是这个硬件的东西，那我们要怎么去优化这个事情，我们就有一个解决思路，就是要实现一个。

Conqueror712(00:18:04): 用户态的孤打模拟器。那这个是就是让用户可以在 CPU 的上面就无需扩大，无需那个 N 卡的一些东西就可以运行扩大 kernel。但是这个运行当然也不是一个运行是模拟，那这个是要模拟扩大的一个 runtime api。目的是就是方便进行测试，然后就是你要写些东西，可能如果你没有一个资源的话，可能不是不会很方便进行测试，这个是在缓解硬件资源紧张的情况下来使用的。大概就是这么一个概念。那简单来说就是这样一张图，我们可以从左边开始看左边这里是有一个。有一个 LAB IS 这个东西。这个是我们怎么的，我们实现的核心就是用 R 来实现的，大家知道现在很多的应用，很多的程序都用 R 重写什么都可以用 rust 重写，那这个也不例外，这个也是采用了 rust 来重写?

Conqueror712(00:19:12): 我们模拟了一些扩大的函数，实现起来了，然后我们通过一些单元测试这里单元测试，然后来测试我们的模拟的函数，执行的效果是不是跟真实的是一样的，或者说比较贴近，这样的话我们就可以去替代。达到一个移花接木的效果，我们这个我们的函数，我们时间的函数会打包到这个动态链接库里面去，因为。我肯定不只是一个线程，或者说不只是一个程序要用这一个函数，所以我们要给它扔到动态链接库里面去方便大家一起来用这样的话我们会得到。这么一个点 SO 的文件。我们再对它进行一些神奇的冒烟测试，其实就很暴力，毕竟冒烟测试你看它没有冒烟就可以了。然后最后我们就通过的话就可以生成我们的输出，然后检查一下我们的输出有没有问题，大概就这么一个过程吧，然后细节有点忘了，我们继续吧，有这么几个模块吧。

Conqueror712(00:20:33): 这是模拟函数的模块，就是我们刚刚提到的这个模拟实现的这部分简单的举几个例子，比如说是这个要获取 GPU 设备的1些东西，比如说是获取这个是完全脱离 API 实现的办法，然后这个是跟 ap I 一起一起搞，就是也就是你中有我中有你。的这么一个方法，这个是单元测试的一些东西，当然这就是简单浏览一下就可以了，然后这个不会放到最后里面，因为这个毕竟是测试，然后这里会有一个专门的 test 的一个小模块。然后有一个非常有意思的东西就是就为什么是使用这么一句神奇的命令来调试，就是我之前是一开始。看了一些文档，然后他说只用这个 C test 就可以了。但是显然如果只能只用 CT 就可以了的话，就也没有这么多话了。

Conqueror712(00:21:45): 然后后来反正就是可能说的吧，可能我们在开发一些，就我们在进行一些探索一些尝试的时候。要多试一些方法就是多试一些，我们可能想到的一些一些一些东西，然后我也不知道怎么表述，就是很神奇，就测试很神奇，就试出来了一个可行的办法对。就是勇于探索吧，可能就是。然后这是我们得到的一个效果，你看输出控制台上就会有我们的设备的一些信息治理设备的数量什么等等的，我们就可以看到了。是一些测试的模块，然后会有一些 logo 出来。简单看一下就可以了。

Conqueror712(00:22:40): 有一个非常有意思的就是它的查找函数的方式和一般的程序是不同的，一般的话就是一般的话直接在代码的某个位

置去进行函数的声明。完了以后去定义，然后编译的时候就会把这些定义去弄，就是连起来。然后那个编辑会生成符号表，然后就 table 我们就可以去通过 table 里面的东西去找到这个函数。我们就可以别的就是在程序别的位置，也可以通过这个来弄。但是一般如果单纯的是这样的话。没有办法很好的去共享吧。然后 C 的话就有一个要它要支持动态链接和多程序都多线程执行。所以说它会搞一个新的东西，就是因为它会在编译的时候会。把它弄成一个中间代码，或者说一些不同于常规的一个东西，那我们就需要去让它来弄一个这样的东西，就是说有一个特殊的符号标示符号修饰可以对应它的线程，或者说对应它的这个设备，因为我们有时候在分布式的，我们在分布式的。

Conqueror712(00:23:58): 那种计算那种算力上去用的时候，我们会要保护它，保护那个代码不会被不会乱跑，然后所以我们要去让它有隔离的一个效果。然后这个时候我们就需要去让它对应起来，就让这个用特殊的符号和它的设备。让它的设备去对应，这样的话我们就可以能正确的找到我们哪个是我们的一个函数，然后这个时候我们去找到才能去正确的得到我们结果，这也是模拟的时候的一个比较困难的地方吧。

Conqueror712(00:24:35): 然后就是总结了，因为我确实也忘的差不多了，就是要说的是什么。我一开始拿到这个项目的时候就是我申请项目，然后我通过的是这个项目的。这个叫什么就是这个项目是通过了申请的，然后但是我对这方面其实可以算是一无所知，就是我当时是大二结束，然后申请到了这个项目，然后但是我一点都没有学过这边包括 rust 语言，我一点都没有学过，然后就是重新学吧，然后虽然说的。

Conqueror712(00:25:16): 学习曲线也是挺陡峭的，但是怎么说学习的过程中还是有一定乐趣的，虽然说就会半夜睡不着这种事情大家肯定也都经历过，不过我觉得就是可能每个每每件事情都有它。每每件事情都能学到一些东西，至少这种根源，这种追根溯源的去调试这种去探索，让我也获得了一些难以量化的经验，就这种东西我觉得很难用一两句话去概括出来，就是可能是一些调试的经验，也有是一些。对于一个项目的宏观的一种理解，就是你拿到一个新的项目，然后你的一个理解可能会更高一个层次吧，然后并且我们我这个我们的社区也非常的负责，非常的优秀？如果没有社区的帮助，我肯定没有办法结项对。那就是大概我要说的吧，然后谢谢大家！

刘晓义(00:26:24): 好感谢龚宇飞同学，然后线上线下的同学们有什么问题，我觉得关于 NVCC 本身里边 vidia 本身里边的一些。对 internal 其实还是有很多很有趣的地方。暂时没有问题，那就是感谢龚宇飞同学，那接下来的这位同学是。

刘晓义(00:26:56): 王征同学是去年和今年都参加了 OCP，然后是非常成功，去年拿到了。是哪个奖项来？大家都忘了，反正就是拿了一个重要的奖项非常的好。对他拿奖不会多给钱的。还是照样给那成给你是我不是用你这个电脑吗？也可以。要不然 slide 发给我黄龙同学去年和今年都参加了通大 SPP，那就是今天就可以连接两次。我本来我不想讲去年的，因为我都快忘了。但是我们的前会长，前前会长强烈要求。这个 IP 是什么 IP，这是在 serve 状态的一个东西。好好酷触屏是不是要改一下，你现在没太清楚？我怀疑你刚才说。对我腾讯会议确实没开共享。稍等。

刘晓义(00:28:36): Ok 来扶着吗？这个家在里面，那不管了，我就扶着吧，那个大家好，我是民族，我本来想先讲。我本来想先讲去年的项目，不是我本来想先讲今年的项目来着。因为我去今年那个项目准备的资料比较多，然后去年项目没怎么写东西。结果想了一下，如果先讲今年项目有点虎头蛇尾，所以我临时决定改成先讲今年去年项目了，去年我们那个项目是叫做清华大学网络学堂助手更新迭代。我想问问大家用吗？不是根据我的身边统计学调查，好像用户不如 LEARN X 多。这还是手机用这个电脑用户还是用电电脑用户好多也会选择直接用网络学堂的一个主要问题是被科目的对。

刘晓义(00:29:39): 对的，因为我刚才就想说来着，虽然我们去年这个项目名义是这个更新维护，但就是我们的千千会长还是不想维护了，然后找一个新的维护者，于是我就来了，然后。从那之后，我也稍微有在继续维护，但是也在咕咕咕状态。先因为当时那个写了一些东西，然后后来结束之后我又。方负责他的一些维护开发功能，所以我现在有点混了，到底哪些是 OS PPT 写的，那个是后来写的。所以我印象中在 OS PB 期间大概就是做了两个部分，一个是把它的工具链翻新了一下。他原来用的就是 web pack react 那老一套，然后把它全都踢掉了，换成了我们更新更快的 wait 和。Tsw vtx 之类的东西。然后新功能的话，其实做的不是很多，因为我好像当时前一个月还是多长时间，好像一直在焦灼于那 redux 怎么给？给他干掉，因为他虽然是用 typescript 写的，但是在 VS 里打开会报一万个 error。就是各种类型错误，我一开始一直在改。试图让他能够至少 link 的时候不报错。然后花费了很长时间。值得一提的是，他现在仍然会报错。因为也是 learn lab 那边改的一些东西吧！

刘晓义(00:31:15): 然后功能做了非常革命性的暗色模式之前。对网络学堂现在也不支持安四模式也晚上打开也会非常瞎眼。然后做了28n，

还有导致我们插件被 chrome 踢掉的原因，其中之一是这个 mv3的问题。当时他因为被踢倒了，现在是回来了，做我们做完 mv3之后。我记得当时踢掉原因好像还是为什么 type 的什么问题好像当时就修了，但是 mv3他当时在2022年被踢掉之后就没修，然后直到去年 OS P 项目期间才把它修了。

刘晓义(00:32:06): 然后除了这些外观上更新，就是在内部那边做了一些新的 API，像是提交作业和获取教师端作业列表。但是非常遗憾的是，这两个新做的功能都没有用户，因为他没支持，然后。Vrx 的提交作业那个接口也是自己写的，不是用的。所以非常尴尬，做了两个单元测试在用的东西。他获取教师端其实有在用，用小。我有点印象，因为有小条子，然后去扒那个作业对确实有不是图形化工具。你你要想扒的话直接用写也可以。

刘晓义(00:32:58): 然后 OS P 结束之后，就是进入了长期的摸鱼状态。那个 learn 那边最近包括前段时间写的功能还不少，包括网络学堂这学期。去年好像是更新的什么收藏备注功能啥的，然后今年好像也加了啥来着，不加了什么公告，过期时间还有什么补交作业之类的功能？不过好像我也没看有哪个课程在用。反正我们做了。还有什么比较搞笑的，有一个2022年有人想要支持课程问卷，但是由于不管是 harr y 还是我们的课程都从来没有人发过，从来没有老师发过课程问卷，我们也没法做，然后直到上周我们有一个课，终于有老师发了一个课程问卷，我就光速。做完了。我们只主要工作就是在8A PI，但是 UI 那边我在写那个新版本，但是就是咕咕咕了吗？现在写出来那个新版本，它只有框架，没有数据。目前倒是可以在这画个饼，目前计算计划打计划要写的包括好久之前就有人提。学期两个学期临近，下一学期临近的时候，可能不同的课程都要发东西，怎么显示把它们混合起来显示的问题还有。像这个什么后台刷新，还有什么网络平台新功能之类的，这个都是画饼之后要做的结果也是哭了一年多了。大概去年的项目就可以有这些风险吗?

刘晓义(00:34:45): 讲讲今年项目之前我们先 QA 吧！说明大家使用的体验都非常好！对你说我想起来了，现在我觉得一个体验很差的地方，好像是提交作业的时候，他不是用的 frame 去网络可以到页面交吗？然后交完之后他意外上也不会显示你这个作业提交完了，这感觉有点难受。大概之后也会修，主要是很难那边对这是一个很复杂的问题。对其实这也是好像大部分人都是直接交文件交附件。那我们再讲一下今年的项目吧，今年我们这个我项目叫 rust 基于完成的异步 quick。对我们最后做出来这个成果叫做 compute quick 不这个名字读音没人说都行，就是 CO 或者读 compute。都行。他做的工作就是他标项目标题应该分为两部分，一部分是基于完成的异步，这个指的是 comput e，然后。

刘晓义(00:36:17): 最后最终我们就做出来了东西干，首先它是 R。其次，它是和现在 R 的社区中写的各种 quick 实现不同的地方是它用的是我们这个叫基于完成的异步的运行时叫做 compute。它实现的 quick 特性，基本上就是 rfc9000的包括0R DT，然后再就是 fc921的 data exten sion。然后才有 htp3，不过这个 atp3的直接用很难用它，然后那个微软现在有一个就是 CIO 的衍生项目，它里边实现了完整的 http3客户端支持就是用 compute quick 做的。然后他现在不支持的是一个是 mod pass，因为这个也还没进 FC 吧，然后另一个就是可能比较罕见的情况就是用户发起的主动连接迁移，意思就是用户在用一个 end 的。或者 connection 用到一半不想用，原来那个 socket 想换一个 socket。就是这个情况，现在还不支持。但是如果是 net rebounding 或者说远程的服务器发生了那个遍地值的话，是可以支持的。然后他提供的 API 大概就是长这样有 connection，然后还有双向或者单向的 stream?

刘晓义(00:37:51): 觉得我在这顺便说一下吧，就是一个很奇妙的事情，就是虽然我们是急于完成的一步，但是我们这个 quick 模块提供出来的 IO 接口并不是急于完成的。它仍然是很接近于传统的定法，因为之后可能会说。介绍实践之前我先超简短介绍一下 quick 和 compute 吧，因为这个里边有些细节跟我之后说的一些时间细节有关系。为什么他叫超级的介绍2.0是因为半年前那个项目介绍的时候没有做过一个超级的介绍，所以我这个是2.0。

刘晓义(00:38:34): 这个 quick 它是一个基于 UDP 的传输协议一般工作在用户空间。但是不过最近用户太这件事情变越来越迟疑了，因为有人想要在 linux 内核里添加 quick 的协议的支持。它的设计目的是为了取，可以说是为了取代 TCP 加 TLS。比方说那个基于 TCP 的 hp2把它迁移到 quick 上就变成了 htp3。然后其他 T TCP 有的功能 quick 也有，而且实现的更好。TCP 没有的功能 quick 也有比方说 TCP 里边你要写多路复用的话，你需要设计一些应用层的协议，比方说 Y max 或者 htthttp2的多路复用，到了 quick 里边它就。自带的多路复用，所以 http3里边就不需要再单独设计多路复用了，然后它也解决了许多 TCP 连接上 TCP 设计上臭名昭著的问题，比方说 NN 次握手这样的。在 quick 里面，我们只需要1R TT 甚至0R TT 就可以建立一个连接。然后因为它工作在用户采导致不同的软件服务或者应用程序可以选择不同的 quick 实现结果就是这个图有点小，我看我能不能打开那个大的图。

刘晓义(00:40:13): 听错了。对！现在结果现在有这么多个不同的 quick 实现他们之间还不一定互相兼容，然后这个网站就是。专门在测各个

quick 实验实现之间是否兼容的。据我所知，这也不是目前所有主流的 quick 实现我知道的就有一个这个腾讯的 T quick 没有被加到这里边，原因是因为他们这个 runner 是用的 github workflow 的 matrix。去跑的，然后他这么凉凉的跑，如果那个 matrix 最多一个 matrix 最多起256个兆，如果再加一个实现进去，会导致超过 gb 的上限，所以就没加。现在是 TC 那边自己在跑 C 去测。我们实现当然也没加进去，因为我们实现的不是很有必要加。

刘晓义(00:41:22): 然后目前 russ 的社区中有很多，也有很多 quick 实现其中最流行的是这个名字。这个它支持 tokyo ssd D 这些，除此之外，第二知名的是这个 cloud flare 的叫做卤蛋饼。他据说这个东西只有 CF 自己的网关在用。没有用户。但是它在 git hub 上有十几 K star 比这个要多。它是 io free 的，也就是说那个用户可以自己用户需要自己写一个 loop，然后处理 IO 处理。它有个最大的问题是它只支持谷歌的叫 boring ssl。然后因为 OPEN SL 它 quick 的支持添加的很晚，结果现在。

刘晓义(00:42:12): 大家的这个 quick 的 TLS 都是百花齐放的状态，而且这个 boring sl 在 russ 的生态中会它和生态有点八叫什么八字不合，他在链接的时候会报各种错，就是和原生的 OPEN SL 会有链接符号名的冲突，然后可能因为导致他没人用。

刘晓义(00:42:34): 除此之外，还有其他各种事情，巴拉。然后我们这个搞的这个东西叫做 compute，它是一个叫什么基于完成的异步运行时？它的出发点可以说是 I，然后在 windows 上支持 IOC P 不使用某个臭名昭著的未文档的 API。然后在其他不支持的平台上，也可以换成普通的非阻塞。值得一提的是，这也和我们之后遇我在开发过程中遇到了一个非常关键的问题，相干的地方就是。我们基于完成的 IO 操作。这个有一个他不说是这样的，它原来基于就绪的 IO 做法是，比方说我们读一个文件，我们是先等待这个文件 readable，然后再执行这个最后的 read，它是一个立即完成的过程。这样的这个 buffer 只有最后在用，所以我们只需要传一个可变应用。但是在这样一个完成基于完成那一步的时候，我们是直接把 C 交到运行时，或者交到内核里，然后 buffer 也要传过去，所以它。设计上是要把所有权传给过去的具体原因我不多说了，大家可以去看半年前那个 compute 介绍的时候的录像。

刘晓义(00:44:00): 我一开始的思路是想因为像 queen 它是支持切换运行时的通过 feature flex 就是支持 TOKYO SD 这些我一开始想的就是能不能直接给 queen 加上 computer 的支持，但是？我试写着发现有问题。他 queen 的就是做那个 runtime 的 treat，它有一个部分叫做 up sock et 这个东西，它要求你去提供一套那个就是提供一套基于又继续基于就绪的这个接口，比方说它。这个 I create io 就是提供一个函数去等待它的，它提供的这样一个读写的函数也是期待你去立刻完成，并且也不给你所有权，只给你一个引用的。然后这和我们的设计是完全冲突的，所以这个思路最后被放弃了，也不是最后被放弃了，很快就被放弃了。但是好消息是这个 queen 提供了一个子模块叫做 queen photo。它包括的是 quick 协议里边和 IO 无关的一个状态机它里边是没有 IO 的，所以这个东西它也没有 event loop，所以就是用户在用的时候要自己去写一个事件循环，然后传递各种事件，然后进行 IO。然后。虽然他就是很适合我们这个情况，但是也没有特别适合我们，最后还是遇到很多问题，但是有这个东西肯定比从零开始要简单，所以我们最后写的这个解决方案。

刘晓义(00:45:45): 就是用这个东西写的，因为相当于 queen，如果是说相当于 queen portal 加上一些封装，再加上 tokyo 我们这个就是加上封装，再加上结果就是。这个就是我们提供的 API 和 queen 基本上是完全一致的。但是也有些更改的部分，主要是我看他 API 有些设计不太好的地方，根据个人喜好进行了调整。

刘晓义(00:46:16): 然后讲一下我在这叫什么，在开发过程中遇到各种神秘问题。我前面好像说了我前面没说就是 queen 它除了 proto 还有一个那个叫做 queen udp。这个东西，它提供了它提供的功能其实和 quick 关系不是很大，它实现的是一些 udp socket 的。比较高级的可选功能，比方说这个 generic segmentation of load 还有什么这几个东西，反正就是就除了这个 E CN，它是在 quick 标准中有提到，可以做一些可选功能的。虽然这个东西。它完全是可选的，但是一方面它确实是目标是提高性能，另一方面我在调研的时候看了现在大部分的主流实现都有这些。就是这几些功能，于是我们又也做了。但是选择做这个东西它其实挺复杂的，因为它需要通过 receive message send message 的那个就是叫什么？Message handler 这个结构体里边去传一个字段，然后来处理所谓的这个叫 ancillary data 辅助数据或者叫控制信息。

刘晓义(00:47:46): 然后这个东西在你到解码的时候和编码的时候，也要调用一些 LAB C 里的各种奇奇怪怪的还不是函数是 macro。这个东西就是调用起来就比较头疼，我们那个 rest live C 里边这些东西也是定义成了对编译成什么来着？我忘了，反正他在 FC 里边不是函数，而是一些木马。但是问题是什么？就是 compute 之前做的 socket 读写在各个平台上的用的 CC 不是统一的，比方说 lion 上它用的确实是 re ceive message message。但是比方说在 windows 上，它用的可能就是直接用简单的 send two 这样的东西。然后我本来想先统11下各个平台的，但是那个 PR 后来还是 close 掉，我们最后选择的方案是去添加新的 code，然后也把那个辅助数据的支持加到了，就是 compute driver 和 compute net 里边。

刘晓义(00:48:55): 不过这个东西为什么非常难写，是因为第一我们第一次写的时候就是有问题，review 的也没有看出来，然后就闷了，然后后来我发用的时候就是在调试的时候发现有问题我就开了一个 PR 这个修。修了之后又没有，又没看出问题来又没了，然后又然后后来又发现还是有问题，又在最后我们最终那个 quick 功能都做完的时候，那个 PR 里边才把它真正的修好。原因的话，它在不同平台上你要用不同的处理方式，你像还有 ICP 都有各种各自不同的处理方式。很蛋疼。

刘晓义(00:49:35): 另外还有一个非常神秘，现在是我们这个实践里边最大的一个问题的地方就是 connection 提供了这么一个函数，叫做 power transmit。他东西你看他接受一个当天时间接受一个最大数据包的量，然后再接受一个八份的课编引用。告诉你你是不是要传东西出去，然后如果要传把东西传到哪，怎么传之类的，这个东西看上去人畜无害，但是就这个东西它引起了非常大的问题，因为是这样的，在 queen 的事件循环里，它的逻辑是这样的就是。先在先等待 socket 可写之后去调用这个 power transmit。然后如果要发的话就直接写写完之后，你还可能如果这个 socket 仍然是可写的，就再泡一次。这样的话，他只有在 socket 的时候用一下他的 bug 就是他在循环的大部分时间 buff 都是拿在自己的手里的，但是我们不能这么干，因为我们的 send 不光我们的 write 不光要交出所有权去，而且他也不能保证。像 socket right 在这个 socketright1样立即完成。这就导致在事件循环里相当一部分时间，我们甚至没有一个能用的 buffer。

刘晓义(00:50:58): 所以我们现在这个 unloop 的逻辑很奇怪，就是他在启动的时候或者空闲的时候去，就是有 buffer 的时候去抛 transmit，然后如果有东西要传，就把这个八份交出去，然后在等待写操作完成之前，我们没法发东西。这导致可能有各种各样的问题。我之前也想过实现一个简单的 buffer pool，但是效果导致性能下降了，可能是实现不太好，导致它的维护 buffort overhead 大于收益了。但即使如此，我们的现在的性能仍然是相当可以的，我们下一页会看。compute 的 the wrong time 现在做这个 buffer 的支持，但是现在还在。还没做完，然后之后会它的设计目的就是里边也是有 buff 的就是在 a 以外的地方做一些朴素的八卦库，然后直接用 a 之后应该会取得一定的提升吧。大概我们现在的性能是这样的，就是因为什么它就是在只有在 linux 平台上用的是 lion，然后其他平台像 mac windows 基本上也都是用的跟 to1样的这个 L，所以感觉比较意义不是很大，就是在 linux 平台特殊的地方是在于我们用的 lion，然后。

刘晓义(00:52:29): Queen 用的一个 this call 叫做 receive multiple message，它可以批量接收宝我，所以我觉得这个比一下比较有意义，结果显示目前。还是略逊一筹就是。但也没有差距很大，而且在不知道为什么在这个测试点上比他还快。这个我们也没有 profile，我不知道什么问题，一个可能是刚才说那个 bug 问题，而且另一方面其实也是可以支持批量接收的用就是用它的 music short code 这个东西可以指。提交一次之后就可以一直收就是一次 CCC 实现非常多次的时候，这个东西如果用上的话，性能会提升也会不少，但是我们现在也没做那个，因为它和我们的 runtime 也是。需要一些很不少调整，大概之后也是可以再改进性能。最后聊一下参加今年 OS P 的心路历程。

刘晓义(00:53:34): 4月份的时候，梅伦当时讲 computer 的时候我就在现场听的，他当时就提到他说他今年准备。准备在 OS P 里进行开展的项目，然后我非常感兴趣，准备就准备参加，然后5月份6月份跟他联系一下，然后就参加了。然后7月份的话就是前一个月，当时就在主要在搞移植它的这个 udp socket 的高级功能，然后8月份就。把剩下的全写完了，然后我们在9月初就末日了。结果就是9月份，一直在摸鱼。然后结果本来是什么最快完结的项目，结果前几周有群友发现了一个神秘 bug，就是如果在连接过程中把你那个 network interface 当掉，比方说拔掉网线。

刘晓义(00:54:34): 或者禁用网卡会导致那个 connection 无法正确的 time out，而是会在你这个写数据发不出去，那个导致可能比方说那个你要开 stream 不断开 stream 导致超过约束的 stream 上限默认是100。会在这种时候出错，而不是正常的30秒。然后结果修了这么一个 bug。然后虽然不是在 OS PP 的范没交到 OS P 范围内，但是这样我们就不是不再是最早完成的项目了。但即便如此，我们好像也不是最后一个完成的。那大概他们就可以讲这些内容！我们再 q1下。

刘晓义(00:55:36): 反正也不。这个我们当时开会的体验就是到了中间的时候，有一部分组还没有开始写，有一部分组已经说写完了，然后并不出现在这个每周的例会上。很僵硬，太勤奋了！然后下一个是我来讲，然后我要现场下载。吓得我都死了。

刘晓义(00:56:37): 不是骗子的电站 download this raw file。一点不能放在皮克斯，至少有一点？它可以不是80P S 文件可以是纯文本。Call script. Post 是源文件，编程语？这是可以让我写写画画的软件，我看看能不能写写画画。对的我可以谢谢华浩 ok，那就是 hello 然后我是代替杨新瑞同学今天来讲一下我们这边的 CRC T 编译器的电路划分。

刘晓义(00:57:50): Arculator 仿真并行化这个项目。然后是因为我没有办法代替杨新宇同学就是 a 宝去讲他的心路历程，那我只能大概讲一下我的心路历程，以及我们这边的技术细节。所以主要都是技术细节。

刘晓义(00:58:10): 好，然后我们回忆一下这件事情是什么？这件事情是说 arculator 是一个基于 circuit 的电路仿真综合器工作原理是直接把硬件表示变成 LMI。对它就是比 V 之类的，要先进一些。V later 是先把硬件 HDL 语言翻译成 C 语言，然后再用 C 语言的编译器，比如 C，再把它编译成比如 L EMI R，所以相当于多了一步。然后就比较的低效 circuit 这边就它可以做到的事情是直接在内部转换，然后直接下降到 LML，所以这个相当于是另一个后端，然后就非常的好的理想。问题是在这次 OS PP 开始之前，oculator 只支持单线程。所以本项目的目标是很自然的为其引入一个多线程支持。事实上证明想的是非常简单的，但是实际做起来。虽然说我们最后的做法和我们一开始的想法是一致的，但是这个主要是受到 arculator 本身和 arculator 打架，然后和 LOM 打架占了这个项目开发的主要时间大战 LAM，然后什么打错一个字。输出几万行的错误日志，这就是 C 加加配合 table 本项目最后完成的。

刘晓义(00:59:41): 事情是两部分，第一部分是我们引入了一个新的 L 结构，这样的话我们可以表示划分出来的任务，然后以及任务之间同步关系。这个是一个相当于比较 infrastructure 的工作，是我们相当于在 circuit 的基础上，我们允许他去表示一个并行结构之后。在这个基础之上，我们去做了电路的划分，就是我们相当于把划分成把电路划分成了我们刚刚说的这种任务，然后我们又去添加了一个额外的 pass 这些任务在后最后的后端输出的时候。把它分裂成不同的入口，允许在运行时外面的运行时进行不同的调度。比如说普通的一个 arc 的输出。你在顶层模块叫一个 model，它会输出这么一个东西，它有一个 model story T 里面存了各种各样奇怪的 SHADOW S，你的 IO 都在里面，它还会生成一个 json。然后它有一个 python 脚本，然后会把读 json 文件去生成 C 加加，这个也非常有这个 LV M 的感觉。也不知道为什么 LV M 就这么喜欢用各种各样奇妙的脚本语言去生成 C 加加文件。很妙。然后这个是如果你的顶层的那个模块叫做 model 的时候，它生成出来的这个函数有点类似的顶层生成了 class，然后你也是1样的。然后我们最后干的事情就相当于是把它分裂成1个 task 一个 T，然后 task 一个可以。

刘晓义(01:01:21): 好，那这个就非常棒！我们首先说一下刚刚说的同步上面的那个 iron structure 引入的这个 infrastructure 叫做。事实上，经过了好几次迭代，我们一开始叫20点，然后二点点 think 觉得不美，然后我们现在把它改成20点 T。事实上是比我们一开始想法更 generic 的一种，就是我们一开始的那种同步的结构是只能用作多线程划分，但是现在这个就是一个比较广义的一种同步结构，它是干什么事情，它是。你每个 task 里面都可以做好多事，每个 task 里面一定是纯的。只有一些例外，是它能够读写 state。

刘晓义(01:02:19): Ok 所以那个 task 里面都是一些纯的东西，但是你可以通过二十点四点 read 和二点二十点四点去读取一个 state，然后有点像一个指针。

刘晓义(01:02:30): 除此以外，task 就遵从 PC 内存续，所以就是你看到的 read 和 write 就是顺序发生了，你相当于把这个 task 都打开，然后都消失掉，那就然后我们在那个名字相同的 task 我们会进行合并。你相当于在前面有个 task 在后面有个 task 他们名字一样，我们会在编译的过程中把它合并起来，然后静态检查。中间这个东西会不会导致它的出现返回违例？是通过我们去扫一下这个 task 里面所有的去完成的。静态检查就是排一个偏序，你直接去看某一个东西，比如说这个 task 必须大于这个 task，那么前后后是不能合并起来。

刘晓义(01:03:23): 相同的 task 名字在我们这里的设计是最后要分配在同一个块上，最后要分配进同一个线程。所以就是我们在这块标记名字，然后我们去做动态检查。当然这个另一种想法是我们完全没有任何的名字。然后但是我们在启发式的去合并一些相邻的框是另一种划分，我们是从顶至底的你当然也可以有另一种从底至顶的，我们的。我们也有想法，也在 PR 里提了，但是没那么做好，然后 question 是说什么？State 是什么？是一个很好的问题，然后事实上在 arc 下降的过程中有很多，然后说到这个就开始怎么和这个 arculator 去打架了，就是在下降的过程中。电路的状态经过了这么几步变化就是第一步是 semantic register 就是你看到的是一个电路上的 register，还有一个时钟，还有一个 reset 值，还有一个。反正就是这些东西 delay 这些东西它是 semantic 的在下降非常早的时候，它就会降低一个东西叫。

刘晓义(01:04:36): 他说这个 S 你就只能用写，所以事实上在 arculator 里面非常早的时候，你的电路语义就丢了很多。然后再往后就越丢越多，丢到了一个地步，它就会变成 LOM 改成那个 PDR。所以我们事实上我们这个 partition 在做的就是二点 task 能够 handle 的，只不过中间这一小块。它既不能太早，也不能太晚，这中间一块比较在流水线比较核心的这一部分能够工作，所以它事实上从某种角度上来说是一个辅助性的结构。

刘晓义(01:05:13): 刚刚说了一下关于 task 本身的设计，我们现在来说一下 task 要怎么划分，然后以及 task 本身所谓的副作用，就是它那个缓存的读写会有什么样的问题，会有什么困难？然后会怎么限制我们做划分，这是一件很不幸的事情，就是我们一开始写了一个实现这个实现 perfectly fine 工作，然后把那个 PR 交上来的时候上游说你要不要等另一个。然后我说好把那个 PR 写完了之后把整个 sata location 给

改了，然后相当于把我们 PR 给扬了。所以我们从10月开始重新做了一遍703里这个是新的 oculator，我们就不提老黄历了，这是新的 oculator 的 sata 的分配方。虽然下降，我们看一个 fertile 的 register，它叫 C，它有一个比特，它的始终是 CLK。然后在某一个时候，我们往里写这个。这个就是福尔特的一个，这个 C 现在可能是一个组合逻辑，它会被降成什么东西，它会被降成这样一个东西，它会首先它会分配两块。

刘晓义(01:06:27): 一个是 C 本身的存储，一个叫 old。what is old cock? Old clock 干的事情是每一次医保的时候就要把 old clock 和新的 clock 比一下，如果不一样，就意味着你看到了跳研？然后你就更新你的 data，然后你你更新完你的 state old clock 就要更新成 new clock 非常的暴力，看起来很暴力，然后显然有更好的做法，就是你可以不存 old clock，然后你通过别的方式去看你这个 clock。

刘晓义(01:07:02): 是不是发生了变化，它的好处是什么？好处是7703引入了一种新的状态下的方式，它会把每个值跟踪一下是。如果我把这个 evaluation 这个瞬间看成一个延，是在这个时钟延前面还是在这个时钟后面的值，所以事实上我们想看 clock 在有没有变化，我们事实上想看的是我们当前这个之后的值。而不是 eva 之前的我们写进的是我们 eva 之前的，所以就是 clock 和这个 C 之类的，它的值之类，各种各样的值都会有。不同的在这个实现它叫 face，就是不同的生命周期一样的一个概念。然后这件事情就麻烦了，这件事情就麻烦在哪个地方？就是说你有的时候因为我们考虑一下在这里写进去的这个值。我们已经写到这里了，写进去的是在我们这个更新之后的值。但是我们后面可能还会用到更新之前的这件事情就会有问题，如果比较朴素的做法是说。

刘晓义(01:08:09): 你就不允许这件事情，然后你想办法去解决这件事情，然后通过电路上，比如你建好多临时存储。然后你把这些东西扒出来起来，这个是7703之前的方式7703之后是什么方式？7703之后的方式特别的暴力就在 eva 的。如果他看到一个值被写进去了。他干的事情是在 evaluation 这个函数一开始搞一个 SIC，然后把这个值读出来，然后你以后要用到这个 register 的时候，他直接用上面早就读出来的保证是旧的值的地方。这个相当于让编译器帮你。

刘晓义(01:08:46): 去做了 C location 就是他自己不做 location 的这些东西就是编辑自动，然后到你的这个站上面，然后就他就 don't care。这件事情对于他们来说是工作。对于咱们来说是不工作的，是因为如果我们已经划分成了 task 之后，你显然不能说一个 task 前面在一进来之后读一个值，然后在另一个 task 也用这件事情就不太行，然后你也没有一个 SSA 可以跨越 task。要求你的 site 的读写必须在 task 里面，你才能保证是没有副作用的。你不能够说副作用跑到外面去。所以我们怎么做的，我们做的方法是我们事实上有点像比较显示的做了 SSA 的那个就是说如果我们对任何一个 CSL 的更新。我们都会建立一个就是我们首先去观察是不是有别的并行执行的 task 会去读的话，我们会在原先的写入地点去把它改成写进一个，然后并且整个 eva 的结尾，我会插入一个新的 task，就这个 copy 过来，然后因为是 eva 的结尾，我们就保证所有其他人都已经读完了。这个冰雹会在这个结尾的这个 task 之前插入同步，所以我们包括后者这个叫 S task，就最后执行差不多是这样一个状态，如果我们把它比如说拆分成三个线程。

刘晓义(01:10:16): 如果一号线程的主线程首先进行 IO，然后产生三个执行主要的计算工作，它会把所有写进去，可能会竞争的值写进，然后不会竞争值直接写进那个 register 自己的所有值里面，我们进。同步完成了之后，会把 shadow 里面的值给写进 storage 正确的位置的值。然后山豆只有自己这个县城可见，所以它就没有竞争问题，非常的妙。

刘晓义(01:10:46): 然后这个结构就是用 task 实现的就是我们看到的是123123分别是一个 task，然后根据我们刚刚说，task 那个依赖关系是通过 PC 上面的 state 的读写去看出来的，怎么能看到一会依赖一？二性和依赖23性和依赖三，但是因为我们在这块去打是通过杀毒的时候打断了123之间的依赖就123变得可以并行。i think I think S 之间也没有依赖，是因为他们都是自己读自己，写自己，这样的话就相当于我们通过这种方式把它的依赖全部打断，然后这两图就可以变行起来非常的好。

刘晓义(01:11:25): 最后说一下 partition planning partition planning 并没有意思叫做我们通过 SSA 可以爬出来状态之间的依赖关系，然后我们按计算器分配就是我们每一个线程会负责一部分计算器的计算和更新。两眼一闭把他给交给麦，然后就说啥是啥？这样的话就是我们生成一个 partition plan，然后我们根据 partition plan 把电路整个里面的各种逻辑抽成 task，然后 task 最后在后端输出的时候，我们现在是比较暴力，是直接把。

刘晓义(01:11:59): 一个 arc arc model 它是现在编译器的最底下是 arc model 会变成 LV M 的那个 function。所以我们直接把 arc model 给复制，像刚刚说就是复制六份，然后我们在每一份里面把所有其他的东西都删掉，剩下的东西就是靠 D CE，它会自己的后面就没有。我看那个 arculator 的流水线，它每一个流水线跑完一次之后就会跑去 CSECSE 就是 common expression，然后并且自带一个 D CE，所以基本上就

是他们也在干类似的事情。就是现代编译器，浪费你的 CPU 周期帮你做 D CE 好！最后的结果是我们跑了一个比较大的东西，我们的 rocket 可以正确跑的就是 rocket big，然后。那个模块，所以我们支持了 memory 然后我们支持了 regis，我们不支持暂时没有支持 T。所以就是你还不能够？可以把里面的值给拉出来，然后去看里面的值，我暂时还不支持的东西是还包括一个是 DPI，但是 DPI 在具体怎么工作，这件事情也是刚合进去，好像大家还没有太说清楚，所以就是先搁置部署搁置不说的语义可能放在并行里面会有比较麻烦，有可能需要加很多同步。

刘晓义(01:13:16): Anyway 然后我们最后的结果是我们的性能略逊于 baseline，就是略逊于没有划分的单线上，但是因为我们的划分算法没有做细节的优化，我们是瞎划分的。所以我们不要看第一行，我们看后两行就是至少这个并行执行还是比串行执行要快的快些，就是非常的好，至少没有说我们划分出来，这个并行执行并没有串行执。那就太僵硬了。然后我们的 PR 是7650好，谢谢大家！然后对最后说一些心路历程，基本上就是和 LV M 大战 LV M，然后每天都在看 a 宝大战 LV M，然后 a 宝睡了，我就在大战 L。基本是 LV M 那边。

刘晓义(01:14:10): 就是。当你会用 LOM 的时候，LOM 很好用，但是当你仔细想的时候，比如说你完成了今天的工作，你很开心，然后你去洗澡了，热水冲热水打开，然后热水浇到身上的时候，你就想。为什么要为有一半是传值，一半是传引用的，然后他也没有说明白哪一半是传值，哪一半是传引用就是 the operation。

刘晓义(01:14:37): 是传的是指针。但是如果你是一个 table generation 出来的 MLIR 里面的 operation，它里边 body operation 指针它传的是值，而是一个类似智能指针一样的东西。但是他又继承 operation 是不是很妙，然后它的等于号是 operation 星的等于号，这个就非常的妙，然后问题就在于，如果这么瞎搞了，没人知道你的那个什么，你的值是谁管你的 value value 它是按值传的，但它里边也包了一个指针。所以它里面就是内存谁在管内存不知道谁在管的问题，就是我们刚刚说了我们那个抽这个方法是拉出来一个 model，然后我们把其他东西都删掉。你把其他东西删掉的过程中，这件事情有问题，因为电路里边的东西是会打环的。把环的话，你是不能把 operation 直接删了的就是它里面每个 operation 都在别的地方有用。所以你没有任何一个地方，你可以抓出来一个值，说值没有用，然后你从它开始删东西，然后它 MLL 里没有支持，说我把一堆东西也一起删掉，然后他们就没有互相的循环依赖之类的，所以就是。

刘晓义(01:15:53): 总之。我们最后的结果是让内存漏出去了，就是 LV M 编译是我占两个 G 内存，然后我编译 LAM 的使用 LV M 编译 LOM 开一个零杠四，然后再连接的时候就把我电脑内存打爆。说明大家都是这样，就是当你的内存漏了就漏了，反正总有操作系统帮你兜底，非常的好好，这个基本上就是我的心路历程，然后谢谢大家！

刘晓义(01:16:25): 有什么 question 吗？在线有什么 question 吗？他们两人都可以打的比较多。对天镇听说就是你，你没有人知道为什么它需要整一个贴？他是白天开的 death. IBM 人不想写一个，还想要更高级的？我可以给你们讲一些鬼故事 table 是生成一个大头文件，但是你在不同的实现里面，你其实只需要其中一小点，然后你干的事情是？在你应付之前 define 你要的一部分，它里面是一堆 fifty five 不能增加价，很科学，完全不科学，什么什么点点 CPC 点 INC，这就是你们 C 家人干的事情。

刘晓义(01:17:40): 该用 R 重写了重那就好，好像没有什么 question，那就大概这样，然后也感谢杨新的同学在 OS P 过程中的。辛苦付出，在我摸鱼的时候，他在清醒的写代码，在我清醒的时候，他还在清醒的写代码。非常的辛苦，大概这样，

刘晓义(01:18:07): 那接下一位我们邀请范书的同学来讲一下，你已经传给我了？在哪里了？有吗。安尼打开 present 模式 ok。差不多。大家好，我是范佩，我来分享一下，我做替换性能分析框架这个项目的。主要是经历吧，因为我回想一下这几个月来做这个项目，唯一的感觉就是充满了一片的混沌，所以这个 PPT 也做的比较混沌我。肯定讲的也会比较混沌，首先就是按理来说，我做一个分享，第一部分应该先介绍一下这个项目的简介，那为什么没有那么。因为我想项目简介这个东西，我想就复用一下。mentor 在宣讲会的时候的把那个项目简介给抄过来，但是 mentor 告诉我他的那个 slides 很不幸消失在了一片混沌之中。所以大家就意会一下 TY 是什么东西，然后我来分享，然后虽然斯太子本体消失在了混沌之中，但是当时那个宣讲会是有录播的，我又重新把当时。

刘晓义(01:19:58): Mentor 画的笔又重新拿出来品味了一下，就是 TY 是一个用其所写的就是 this time 的一个就是扩展的加速器，然后我当时听完这个宣讲和看这个东西，感觉像是希望有一个非常成熟的硬件团队，好像里面该有的东西都写的差不多了。

刘晓义(01:20:25): 他们需要有一些工具就是有个 profiler。profile 可能需要点可视化的东西，做硬件的人都不喜欢写 GUI 这个是可以理解的，可能我做的工具，就是学一学 GUI 怎么写，然后就和他们配合。帮他们学习 GUI 就在这个过程中，我能学到一些东西，比如说 RV VR VV 它。那个贝克加速器是怎么设计，另外就是体验一下世界先进水平，因为这虽然是个硬件项目，但它并不是一堆 make file 拼凑起来了

，里面用了 RVV。

刘晓义(01:21:14): Mix 骑手 rust 感觉这四个东西拼在一起，一看就非常的灵，有一点意思参加，虽然我。不会什么前端也对写 GY 没什么兴趣，没关系。可以学，然后最后还说一下我要做的一个东西，这可能是整就是。就是 T one 它只是一个 RVA 的核心，它还它在工作的时候还需要有一个 RV 的 scanner 核心就是。就我们目前用的 rocket，然后配合起来工作就是 rocket 遇到 R 微微指令的时候，把那个指令放到 EQ Q 里面，然后 T one 去执行 T one。执行完了之后通过一个 Q，然后告诉就是 scanner core 那边 retire 掉了，其他的还有一些组件，我做的这个东西 profiler 就是。把整个就是在区所这边加入一些可能加入一些 profile 有关的事件输出吧，输出出来，我把它给。写好当时和讨论预想的事情，唯一有什么问题，就是因为 RVA 是一个比较复杂的东西，希望也是一个复杂的东西。然后七索也是一个复杂的东西，就整个东西非常的后现代，然后就是所以我尝试学习就是那个时候还是在6月多份的时候。

刘晓义(01:23:02): Os pp7月开始前。感觉这个东西学不太懂。然后就和 mentor 讨论，当时对这个东西确实比较复杂，就写这个东西，写 profile 的这个东西就是需要有一个微架构文档，这个他们暂时没有写。但是之后可能现在微架构那边也在做同购之后可能两两个礼拜之后重购完了我们就开始写破坏掉的文档。我们就开始写微架构的文档了，然后你也开始可以开始写了，然后可能到了今天我问就是这个微架构的文档怎么样了，那么这个微架构。确实还在重构，因为我看昨天他有一个大的 PR，大概刚清微架构重构的 PR 刚清理完了，还没有太不知道现在合进去了没有微架构文档，现在。

刘晓义(01:24:08): 忙两个礼拜之后我们就会开始写了，然后中间，你们看到这一个。这一个。这个框架就是我到6月份的时候，这个框架还不是和这个长得完全不一样，那个时候还没有一个还没有把。整个 test bench 还没有这个 rocket core 是一个假的东西，当初就是 DPI DPI 内部这边也是用。也是用 C 加加写的，然后 deep test 也是用在 stack 上面用 C 加加分的一层，然后整个可能整个暑假的过程中，我看一时半会好像也推不太动。就可能做了一些简单框架性的工作。然后就投入到整个7万的项目开发过程中，可大家其实就是这个图中的可能每一部分组件都重构了34遍，然后在这个过程中，我成功已经。

刘晓义(01:25:20): 混成了 T one 这个项目可能部分的就是相关的东西都是我设计的可能唯一的问题，就是到现在整个东西还没有。能够让 profiler 有什么实到实质进展的一步当然到了最后我也是写写了一些东西了，就是把这个框架给弄好了之后，就是我们一开始的技术路线，是说我们用这个用 json 格式的 event 的方式来记录所有的信息。但后来可能考虑了一下。就是把它全部借助成 jason event 这个空间占用可能会有一点倒闭，就变成了把可能需要的一些信号把它给抽到了一个 module 里面。然后我们跑完仿真之后有这个波形文件。一般就是它会跑出来一个 ST 是一个二进制的波形文件，然后用一个工具去转换，就是把这一个 module 下面的所有信号给。抽出来，因为这只是一个比较小的 module，所以说它的体积也不是很大，然后接下来就是用 provider 做一些分析。然后分析，也分析出了一些东西，比如说 RVV 所有指定的发射提交时间，这个东西是拿得到的这个东西有什么用。目前来看可能用处不是很大。

刘晓义(01:26:54): 但在开发过程中还是有一点用的，比如说我记得当时我在写这个的时候就随便找了一个 T。TP 然后跑完发现它的输出是空的，我就非常的支持，这到底是什么原因，是不是我哪里写砸了，还是这个 build system 因为是历史写的很复杂，哪里出了问题，最后经过深入的研究发现。整个 test case 里面它就是一条微信都没有，因为因为 GCC 的自动向量化，它不知什么原因倒闭掉了，所以有一些 test case 里面确实就是一条微指令都没有。当时就对 T one 这整个项目的每一个部分，它到底有多灵产生了一些更新的认识。

刘晓义(01:27:53): 然后说一说收获语音感悟吧，这个东西也是当时的宣讲会上面前面几条都是 mentor 列的第一条就是最激进的硬件开发流程，这个我确实是学到的那个 nx7。这个非常的特别是这个构建系统是 mix 写成的，它体现在两个方面就是本来一个就是硬件的项目用 mix 已经很了，并且它里面的 nex 写的也非常的 S。搜到什么程度。我之前想调破发的时候我想写一些。那我发现那个 mix 它太说了，我不知道怎么在里面加一个新的 test case 进去。我现在怎么知道的这个事情，因为两个星期之前骚的一个 PR 里面加了一个 test case，然后我就可以。就是 display，然后我终于可以照猫画虎照的，知道怎么往里面加 test case 了，然后上古的后现代体系结构这个 T one 确实先进，但由于。里面的代码的命名风格和微架构文档风格也非常。我读到现在还是没有太读懂，最后？本来这个 profiler 写出来，一方面是 T one 这个项目有，另外就是对 sequencer 自己也有用，因为 profiler 写完了。普拍了就可以 profile 出来。很早就找接着找。然后头发出来就可以出数据。然后数据出出来之后，那个论文的 experiments 部分那一张表格列下来，就省了一大份工作量。但是在我们的共同努力之下，有这个 profiler 做的有点倒闭，那这个论文这部分数据如何出就只有请。

刘晓义(01:30:24): 八仙过海各显神通最后。我体验这个过程，我回想这个过程的时候就想到了，因为我写 rust。就想到了 rock 有一个大家都很样的 feature 它叫 specialization 他15年的时候，就有人写了一个非常漂亮的 RFC 但是直到现在，我也没看到他有什么核兴趣的希望处

于一片混沌之中。

刘晓义(01:31:00): 感觉做的也混沌掉了，但是最后来看这个 profiler 做这不还是有一些？就至少做出一些东西。第一，它确实就是给整个 T one 设计的一个 profile 的框架，虽然里面具体的内容就是一些怎么抓微架构的。微微加微架构的信号和设计就是 performance counter 这些东西我中间可能做了一半，然后发现这些东西过于混沌，最后它倒闭掉了，没有合进去，这个。但这个框架本身确实设计出来了，然后也理解了，真的要把这个 profiler 做出来日方长至少还有一些很重要的东西，比如说内存子系统的仿真和我们需要就 profile 的设计，还是需要准备写一些 test case 我们目前的 test case。都很灵，就虽然大部分 test case 不至于零到，由于 GCC 的 aoration 倒闭的一条 RVV 指令都没有，但是也零到就是。不太好用，后面需要做，然后可能等这些做完了之后，等有微架构定下来，然后微架构文档写了之后，然后再一起做性能建模和就是怎么去定义 HTC 这些真正该做的事情吧。

刘晓义(01:32:42): 至于一开始宣讲会上说的那些事情，我感觉有张 PPT 上写的。锦上添花就是以后有时间前面都做了，再慢慢写吧，总体来说这就是我做完整个就是 T 这一个 OS TP 项目的总体的感受好，就下面是 QA 的时间，大家有什么问题吗?

刘晓义(01:33:44): 那就下一位同学？感谢范淑培同学！好，那接下来是我们来有请谷秀海同学来做收官，事实上是我觉得可能是本次 OS P P 里边出纳这边蛋白量最大最辛苦的一组。对用用你是吗？对的，可以，那就用我吧！

刘晓义(01:35:05): 对可以用了。你还可以用完了我没写名字。没关系，我的名字是古华，这个是我们巴巴，是巴士粉丝项目汇报，因为就是。粉丝意思，因为这个项目的话。他原定的目标是在 OS M 中。它道路是这样的，它分为路段，还有路径，还有那个路的 feature。是一条路的话，它是由多多段路来组成的，但是问题在于我每一段的公交车，它不一定是包含整条路的，它是包含有一条路的路段。但是如果这一段路原本它是一条完整的路段，被包含到这条路里面，那么我们。为了画这个公交就需要把这段路给切断，然后让公交包含其中的一段再把路的 relation？给改了，也改成在里面切成一段这个功能的话。我的项目里也有做一点，但是没办法做的很好。

刘晓义(01:36:31): 那么这也那么接下来就是开始正题，我们基于前端技术这样的 open stream map 公共交通关系编辑器。这里主要讲的是几点吧，一个是我们先看一下账目的。半成品的成果因为因为这个项目它的工作量还是很大的，跟竞品有社区沉淀了十年的开发。那个什么沉淀了十年的那个 ID 编辑器有 facebook 赞助的一位全职开发开源开发的人。一直的一个 rapid 编辑器，然后。所以我这个项目还是比较的跟他们竞争起来，这个功能还是很少的，东西也是非常少的。另一段的话我们会讲一下我的开发的重构过程。还有一些我个人的一些感悟。再有一个部分的话就是反思一下，还有甩锅，就我为什么写不完，最后是未来展望。

刘晓义(01:37:36): 顺便给社区画画饼，说不定能写完了，anyway 继续开始。首先，项目成果展示至少我们还是尝试过的。我们完成了一个包含主要功能的原型。不算这个圆形的什么，它什么都挺好的，就是不还好用。不太好用。功能都有你，它可以获取地图，可以筛选数据，也可以渲染数据。它的编辑功能的话。它是可以加入位位置，然后可以进行添加，删除原信息。还有一些基础的功能，它上传导出也是有做完的，它可以导出 J OS M 的叉 ML 在这进行后续操作，为什么它上传导出要依赖另一个软件，是因为 open map 它是有很多实现的。比方说我们这个目标是做一个通用的编译器，然后它不只支持那个 OPEN API 官方的 API D 我们比方说我们还可以用 opengo fiction 是一个虚构的。那个地图就架空世界地图的一个 API，然后他们 open stream map 是设计了一套上传的健全标准。设计比较复杂，它的功能很完备，因此它比较复杂，然后很不巧，在本来实现这个时候我在赶工期，然后再赶别的内容，然后把这个内容我。

刘晓义(01:39:21): 我跟导师讨论。大概十几分半小时间得出了一个结论，我们要不用另外软件去上传，就这样。那么。这是一个 overview，有点小。但是有点小，不知道能不能放大。放不大就算了，因为底下每一段都会有细角。这是 overview 左边的话我们就可以看到这是个地图。我们选取了。一处美国的地图因为理论上不该有地图?

刘晓义(01:40:04): 在上面的话，我们可以看到这是一个一些工具栏上面粗暴的排列了一些实现的功能，因为他们本来可以做成二级菜单，也可以做成右键菜单，但是我都没有时间做，所以就这么粗暴的放在上面。右边是大纲，右边的话上面是一个大纲，视图，就是说它展示了我目前获取到哪些数据，然后我自己创建了哪些数据以及所有数据的一个总览，右下方的话是一个。

刘晓义(01:40:36): Property editor. 我可以把当前选中的元素来进行一个编辑他的源信息，比方说他有哪些 T，他有哪些成员？就是比较原始的使用了 OS M 的数据结构，没有做那种抽象编辑功能。一个是地图渲染功能。这玩意看起来挺不错，有虚线，有实线，有各种主主干道和那个支线的分别。

刘晓义(01:41:08): 为什么他写为什么我写这段的时候在赶工期，他写的这么好，因为他整串的颜色主题都是从友商那里抄过来的？我发现

因为我用的是 PCG S 的技术，然后另外他用的也是 PCG S，所以的话我们用来画地图的思路是差不多的画地图的思路差不多的话，我想了一下，我把。

刘晓义(01:41:34): 东西稍微改了一下，跟它的样式表兼容了，我就直接用了它样式表。得到一个非常不错渲染效果，当然我感觉这也是这里唯一能称到的地方了。以及还有编辑功能大纲和属性。这里可以看到我们以树形菜单的形式展示了当前的关系。Public transport 的关系就是说当前有哪些道路，然后道路分了哪些路段以及下下方还有一个 relation editor. 底下其实还有一段没看到。我看一下能不能把它拖出来。对底下还有一段。这一段的话。它可以改成一个 T tag，在下面的话，其实还可以通过拖动选中来改变其中的成员关系。还有更多的功能的话，这里不方便展示。一个是节点创建，我们可以创建新节点，并且移动它还有可以创根据当前选择元素，创建路径和关系修改关系和顺序导出目前的编辑。看起来挺美好的，但是我们这项目在最开始立项的时候还说了一个功能。就是说我们本来想要有加一个自动规划的路径，规划功能，这样功能的话，其实最后一段时间我要尝试加进去，但是因为因为没有调通就没有放。就比较遗憾！开发。为什么会最后没有时间，我感觉首先第一个月的选型就会出了一个非常大的问题。

刘晓义(01:43:30): 大概的开发流程是这样，我第一步先选了一个原生 JS 架构，然后写个渲染逻辑写的还挺舒服，但是直到我要写编辑功能的时候，我绷不住了，我把它重构了使用 react 重构了一遍。接着再去写编辑功能一直写到 DDL 还写的不是很完整。最开始的话，我们选用了最开始我选择的是 js module。加一个 PTJS 库，然后就开始手搓了原因的在前端里面其实不是很常见，而且不是很高效的一个选择。他听起来很酷。信息有点膨胀了，那个时候还有一点是我看到友商也就是那个 rapid 编辑器，它的技术就是这么写的，它就是用 js module 和那个 PCG S 自己做类自己做一些类做抽象，然后自己整个整体的架构起来的。我看了这样子觉得他能做到，我对着他的抄总能抄出来，结果打脸了，抄都抄不好。

刘晓义(01:44:44): 为什么我说超不好，我在写编辑功能的时候遇到了非常难的问题非常不好解决的问题。一方面的话是全局的状态同步，我看 rapid 编辑器怎么写作的，他写了一个大概有一一千多行的类去解决这个问题。我肯定不太好这么做，因为它有它，这1000多行类的话，其实它还包含了几个分别有几百行代码规模的系统。嘲笑我这一过，我一看这还了得这项目还写不写了，尤其是涉及 undo 和 redo 的工作的话，整体的项目，它的事件。就爆了。还有渲染管理的话，手动管理自己的组件，正确渲染的话就非常难，因为你需要监听各种信号，数据一来就得手动改。

刘晓义(01:45:53): 类，还有一个是类型系统就 rapid 是 JS 写纯 JS 加上一些 js docker 来写的。然后 js docker 有个问题就在于它能标注数据，但是不太能不太，但是不太能就是它会在一些复杂类型推断上和一些复杂类型标注上，在和 vs code 打架和 es link 打架。然后打架结果就是回退到 JS 类型，JS 的类型就是没有类型。这类型没有类型，于是你就需要比方说我要找一个类有哪些参数本，哪些成员和哪些方法我就得。把这个类的文件找到读它源码，而不是在 vs code 里敲的时候自己弹出来，这也是开发效率的问题。我本人能重构的能重构就全部重构的方法把 JS 也全部迁移到 TS 去了。最后就是开发效率。如果我用框架我可能截上就写完了。我自己喜欢的。写了写上100来行200行的。外部同步工作，所以就只能被迫重构了。继续重构了之后的话，开发工作就非常的是，终于回归了现代前端的技术，一个是。React T typescript 和用了工具。听起来是个非常正常现代的其他项目。

刘晓义(01:47:35): 接下来继续看，然后 PCG S 的话其实它也有官方提供了一个 react 绑定。所以其实原本不用这么复杂。我有比如说我第一个月干的工作，用 react 写的话可能两星期就写完了。那么我最后可能就会多算三个星期时间来写功能，说不定就能把那个功能加上。而不是像现在这样就要拖到 DDL，然后结不了就时刻担心自己没法结相。

刘晓义(01:48:10): 粉色甩锅。这个部分是没有达到项目的最高预期，就是说有一个功能它没他到最后还没写完，就没有合进去，到现在其实还也没有时间写，因为它其实是个很复杂的功能，就是说自动的路径规划我。

刘晓义(01:48:30): 这个东西非常复杂，因为它需要跟 OS M 的数据结构和一些在线的 API 打交道，还会涉及一些比方说我选出了多条路怎么办？比方说我像缺德一样绕了远路怎么办？或者说百度和缺德，高德一样，绕了远路怎么？能不能还要加一些 fallback 的手写规则，这是这些流程我通通都没有很好设计，目前也没想出来，很好设计。不过在我们完成了一些基础的原子化的功能，跟很明显的根据数学归纳法这个项目能完成。所有的编辑工作，至于开发体验，如至于编辑体验如何的话。相信我应该能相信看到这一段开发的过程，我们也可以原谅我，并且允许我在以后慢慢的完善它。

刘晓义(01:49:31): 关于技术选型方面的反思就是说。时间有限不能自己造的是很酷，但是时间有限。而且尽量以开发效率优先，就是说轮

子这些东西的话。本来就是为了提升开发效率和幸福感的一个东西。如果自己要追求挑战的话，最好还是不要在这么大的项目上直接开追求。

刘晓义(01:49:57): 以及开发流程的话，项目其实很偏向产品。它需要一套完整的管理流程，单人前之前开发的时候。刚开始开发的时候。早期开发并没有想到这么多。直到后面我才开始写 to do。因为到后面的时候一堆需求压成三了，我不知道先实现哪个，我就把它们全部写成 to do list，然后我再给它排序，说大概怎么样。

刘晓义(01:50:29): 接下来就是一些杂谈。比方说后续的需求就是我们刚刚一直说的完善路径，自动规划以及还有一些编辑功能，提升幸福感的企业。比方说我举个例子，这个编辑器的话，它的持久化是有问题的。就是说。你刷新之后。你的所有变更都会被隐藏好在不是丢失都会被隐藏，你会看你，你会被吓一跳回以为自己东西坏了，但是如果你按下 ctrl 自己撤回一下，就会发现自己变更都回来了。这个为什么，我猜是因为那个。the undo 的组 the undo 的库，它在漏的值之前，不知道为什么先把自己的先把自己默认状态加载进来，并且作为自己原本的状态，而没有去读自己原来的持久化的状态。我不知道怎么修，目前还不知道怎么修，所以就只能麻烦大家按一下 CTRL Z 每次恢复的时候。这是一个例子，这些需求的话说问题说是问题都是问题。但是如果都修的话，估计修也修不好太多了。后续的话，我还是希望能够通过这样的项目，还是希望能够更多参与社区的贡献。

刘晓义(01:52:05): 虽然说这次。

刘晓义(01:52:21): 虽然说这次并没有带来想象中那么完美的作品，但是起码也算是做了一个作品，希望。我以后可以继续尝试维护它。哪怕是能为社区解决一点问题也好。至少我这个项目的话比友商的项目还是好要好那么一点点的，因为我的技术，因为我这个项目的话，我的所有地图绘制，还有那个元素绘制都是自己画的像友商那个比方说就不提 rapid rapid。太强了。难怪 facebook 要赞助他，然后我很不幸就没有达到他的水平，但是另外的友商的话，比方说那个 OS M 它其实是用了一个，它其实是用了现成的地图库，然后在自己在上面加个图层，再在图层上面用 dom 元素去。去画那个地图，我觉得我这个实现怎么样，底子还是比他好一点，后续可拓展性应该会比他好。如果我愿意，如果我能继续开发工作的话，是能打造出比他要好一点的那么一点的编辑器原型的说不定吸引到了，说不定惊动了 rapid 的本子，然后 rapid 的那边就那位老就。

刘晓义(01:53:44): 把我的功能全部抄回去了，虽然说。但这里想的还挺远的，总之我觉得这个项目目前要分享到这里，那么有什么，那么接下来 QA 环节有什么吗？线上一个平时有什么线上。现场好像也没有什么问题，那就谢谢大家！就今天就讲到这里吧！那今天就胜利结束了，然后今年 OS TP 也是非常的圆满，希望明年各位。无论是同学还是导师，还可以继续来图纳这边来参加，然后同学们也可以尝试一下，看看能不能做导师来带一台项目。后会有别样的体验。好，那别样的体验可以从被压榨变成压榨。然后今天也不早了，已经九点，然后非常感谢大家，无论是线上还是线下来参加，然后就这样热烈庆祝，我继续顺利结下！