Core Community Call 2 Notes

Meeting Date/Time: Friday, January 20th, 2022 19:00 UTC

Meeting Duration: 26 mins

Video of the meeting

Decisions

- Program Runtime V2 won't be ready for implementation for a while
- Transaction instruction limit is 64 in an [upcoming feature]

Meeting Notes

Introduction 00:00

Jacob Creech: Welcome all to the second core community call. There are a number of agenda items that we've proposed for this call. Currently, in the list is first going over program runtime V2 with Alexander, Alessandro, and Aditya as well if he is on this call.

Then there was a request to go over the transaction instruction limitations and figure out what makes sense on both the Firedancer, as well as side to implement. Finally, if we manage to make it and have time, we'll go over with the Mango team application account write fees SIMD.

Housekeeping Note: This is for future agenda items.we're going to create an issue for people to propose agenda items on, so that it's not just reach out to someone and hope that it happens so i'll put the link to it here. Ultimately for anything that gets proposed, we should have at least some documentation on it, it should be ready to discuss it. It shouldn't just be an idea.

From there. Alexander Alessandro, and Aditya, feel free to get started on program runtime V2.

Program runtime V2 1:46

Alexander Meißner: Okay, yeah, thanks.
First of all, disclaimer, everything's still a draft. It's not even a proposal yet. You won't find a SIMD for now. So lots of things can change. Timelines are still being worked out, and this is mostly a teaser to get some community feedback. If these are the things, then it's you the programmers on the same thing very well to you if this is something you want to see, or if there are things where you think like oh no, this is never gonna work for us.

Support for the Move programming language

Okay, so the biggest elevation is that we want to support the move by preparing like this. Some of you might have heard of it.We had support for Move at the very beginning..We sketch it, and we then implemented our current trust ecosystem.So in that sense Move is coming back to Solana .But this time we plan on compiling it, and having it compiled and verified at deployment. which is something new. I think I would change so far as I've been trying to interpret us and further beyond that we want to make it an interoperable Delta frontend.For example, with CN where it brings a lot of services. But also some very interesting opportunities.

Typed Interfaces 4:21

Alexander Meißner: So yeah, what will change in the program runtime? Programs will become libraries essentially. All the programs will be able to export types as well as countless constants.All the functions will have a static parameter and return level signatures, no generics on these public functions and every public function will then be able to become an entry point. Currently, if you're not using framework like anchor,well, basically everybody is writing something like a dispatcher in the entry point. They deserialize the incoming instruction and then immediately call Catch up in the extra point and then imaging that call to be called into something like a such case or Imaging that call, another function. So that would not be necessary. But just call into the right function directly. and have it be using a type interface .Also accounts with the contacts.Currently you have to use CLI simply,serializing if You need to know what you're doing.and what your bytes are, how to interpret them and in this new runtime we would essentially give every account types.similar to structure. It is not yet decided how to dynamically size vectors. I kind of work on that, because, obviously, as you know, structure in Rust . See they have static layouts. So there is no concept of these programming languages of dynamically sized in lines of members in stocks.Possibly

you might have to statically declare dependencies.By dependencies I mean the other programs you import from and it's you that use functional stuff.

Finer Sandboxing :7:07

Alexander Meißner: Okay. The other big change is gonna be Finer Sandboxing. Sandboxing, the idea of separating programs coming from different developers so that they cannot misbehave beyond their own little sandbox. It's completely done by virtual machines and separated with our address spaces, we have one for every single instruction and transaction and that is very inefficient. In the current architecture we actually copy all the data in between those sections. We have a feature reading this which will understand the memory map, at least in between these VMS of the accounts. But still the programming model is and continues to be for the old program runtime. i.e, every single instruction gets its own virtual address space. And with this new runtime, we want to change that. We want to make one big,VM and one big address space for the entire transaction and share the course across all the instructions
and instead of the way, find my grants, access control system. which basically does access control allocation of a number of objects. So, you would be able to declare time to make an instance of that time in some data and obviously store it in your own account with retrieve that format, but you could also. That's not the interesting part, but you could also pass it to some other program. And that program could also start in its accounts and retrieve responders and pass it along and so on. And all these programs would be prohibited from modifying the data. So for them, your exported types would be okay. So now you will be wanting to know, what's the sense of exporting types when you can only have opaque types. So the idea here is that if you want to do something with that object from somebody else. you're returned to them, you'd call them and say to them, "Please do this modification" and now as they are the owner of the time. They hopefully then know what should be possible on this type and what should not. So in this sense, what this allows is, you could have assets declared in every single program. So for now currently on chain you have to use a token program to have a token plan .With this model of exporting types which are only modifiable by the program because exporters attack, you could have every program be its own token program in that sense.

With this comes also a change in how CPI works and Cisco's work. So currently CPI calling one program and another program is really expensive, just slow and limited. So instead, we want to make that more like a function call. Essentially, we would be passing the instruction parameters, empty accounts or a small panel that's likely to work in a function code. Cisco would be replaced by CPI to build programs. Some of you might be aware of Cisco as a special mechanism. And that would also be going away and just be treated as another call which happens to be a better program.

Limits and Costs 11: 40

This is actually the least clear from what they expect new limits and costs economy because it obviously depends a lot on the specifics on documentation. But what we want to achieve here is to remove some of the most annoying limits. So one of them, probably most of you have found out, is that you cannot only reallocate accounts by a fixed amount of 10 kB per transaction, and If you want to do more, you have to do a series of instructions and you can come to the only do for CPI nested in your server because that's also means that we have for our VMS suspend simultaneously. That's something we need to keep the memory alive for the other VMS.This is why that's both limited for now.Both of these limits should be gone.

Additionally, a limit is that we have if it's stack and the stack is quickly consumed by the fact that every function call that's 4 kB of section. And we're switching to dynamics. So that the compiler can actually dictate how much stack memory it's going to use in response.Then, hopefully, more efficiently we use the same, which should enable more or deeper function call traces. CPI, as I already said, it is gonna be cheaper. So we will also calculate less you for that. It won't be the patent on the account size anymore. I think that's also a limiting factor for most of you and Cisco might be slightly a bit cheaper. Then you are probably gonna introduce the cost for declaring programs in a transaction definitely. Currently That's it free, as far as I know. because we might switch to EBR reloading programs which is an amazing bonus and there will be one based cost working program or invoking a program independent of if this program is doing anything at all. Just returning immediately, if this is also not the case. Currently there are different costs associated with depending on if this program is a top level declared in the message of the transaction generated by CPI.

Things that Stay 14:50

Yeah, then things which we are not going to change, in other words, which are going to stay. Rust will still be the main programming infrastructure. So it's not that we are moving to move. We're staying with rust. We're just also implementing move for people who come from other ecosystems and want to program on Solana . This just part of the motivation .Bigger part is that we want to print these concepts of move .I think having every function as a possible entry point and having typed accounts. you want to bring them to us.The one pain point you all know about having to be clear and all possibly access accounts and called programs upfront in a transaction is going to remain it. It has to do with how the transaction scheduler and patching works and if you bounce or if

the program on Time doesn't allow that upfront. it gets really hard to do this efficiently. This will remain something which. probably with some of the solution, like simulation of the transaction, to figure out which programs do Well, actually want to call, and then submitting that section. Then, the account model is mostly unchanged that will stay more or less the same. Like the accounts. Database is still few values for that part .The rent field being deprecated. But it's already on their way, and the type 3 of this gonna be introduced.
Now, the interesting thing for you is probably how are we gonna migrate from the current system to the new system? So we plan on doing a long migration period by long, I mean a year, at least, I think, in which you will support both my times and you will be able to call the old runtime from the new one .This gives you the opportunity to to move all your funds into the new programs and in general all these changes are so fundamental that you probably have to rewrite and redefine all the programs completely. So this one big migration movement essentially, and therefore we will only want to do one so that the community doesn't have to multiply waves of redeployments. We want to prevent that and are trying squeeze it all into one pick up there. Essentially.: Yeah. So this is for my presentation.

Okay? Now yeah. For questions.

Jacob Creech: Cool. Thank you. Alexander. Yeah. Being cognizant of time, we have about 8 minutes left. So if people have any questions, especially from the fireddancer side, since you know this was a request.

Kevin Bowers: I have a couple of quick reactions, and I also notice the timer, just instead of asking the question, is going to rattle off a bunch of things that I was making notes on as you were going.

I think, overall, my quick reaction to this, the official reaction. But this is the first time we've seen anything more in detail about it, as there's a lot to like here. There's a lot of doubles in the details, though, that we would have to sort out the most, maybe down in the firedancer code base, the way we structured it is. We actually started from the programming model we put together for pith for running stuff on chain, and then we've extended that to cover X 86, and we're writing over code within that to give us a lot of flexibility for writing both implementations to run stuff on chain, and also do stuff on chain. And , give us some cross-platform capabilities and interoperability with various hardware kinds of acceleration and whatnot we're planning. With all that, like what you're describing nicely maps onto that. So I'm not seeing anything there that structurally impacts at a very low level a lot of our development plans where things look

a little bit more on the logistical side, which is what's the kind of timeframe expected for this? How certain is this? I know you mentioned. This is before not even a proposal, just a thing kind of throwing out there and then, when we're looking at it, we're essentially like. My thing is, should we be targeting to just implement this, or we should have them on both, or should target the old thing, and then wait to see if this happens. And then, you know you already dressed this a little bit. What's the kind of thinking around interoperability costs, and you know some of the details of going there.
I think there's a secondary thing that would be in there. That is also we've done a lot of work on cost estimation. We're spinning up a lot of our internal pot resources to do even more elaborate work on there. If this is going to change the cost models that changes some of the quantitative research. We'd like them to be doing on existing transactions, and so more details on that would also be of interest to us.
I'll shut up now and let up people talk or try to get in the remaining 5 min. What's done?
Sorry for talking really fast there, but I was trying to cram a lot in the interest of time.

Alexander Meißner: Yeah, just too quickly to answer that. So as I said, timeframes are not clear at all. Well, the most optimistical one, is.to have at the end of the midterm something on testing it, but like really really minimum MVP with most of the features missing just to have some rough idea what this is gonna feel like.No documentation, no support whatsoever. Just for the raw code base,the skeleton of the code base essentially And for how committed are we on this? This is actually the spot out of another project. This is going on for 2 years now to reform the existing program runtime. and I think we have reached the limits on what's possible with our present day. Right? So this is all the things we have conducted in the last 2 years along with our ground architecture.and how we want to reform it.
So,I think that there's definitely something coming in this sense, if it will be accepted. This thing which I just presented to you that is still up to the date and the other thing about firedancer, yes I'll most likely expect you. We'll still have to implement our program runtime time because of this long migration period.So we're speaking here. I thought maybe something would be ready on mainnet in a year, and then another year of migration period.So if you only start implementing your program runtime in 2 years, and you might get away with not implementing. But if we want to be faster, then yeah.We would have to have that both.

Kevin Bowers: Thank you.

Jacob Creech: Okay. With 4 min left. As anybody wants to get into any other questions.

Philip Taffet: I raise my instruction limit, request. ‘

Jacob Creech: Go ahead.

Philip Taffet: So some of us talked about this in the discord. But we're interested in limiting the number of instructions in a transaction. because, especially as a transaction size grows. or as the maximum like MTU grows.then you have this problem where someone could put like a 1000 instructions in one transaction, and they're useless instructions. But we don't want to support that because we don't think there's a compelling use case for it.

Alexander Meißner: Yes, yes, this is already independently of this project, we have a feature for that thing already on the schedule. This will be limited to something like 50 or 60.

Philip Taffet: Oh, even better.

Alexander Meißner: Yeah. So we measured what the longest currently used transactions are on the system and he was something around 40? So we thought, yeah.
So something slightly buffed up.

Philip Taffet: Okay, thanks.

So this is a feature flag that has not been activated yet?

Alexander Meißner: Yeah, that's already on schedule.

Philip Taffet: Oh, okay.

Alexander Meißner: If you ping me on discord and in the virtual machine. So then I can send you the ticket application.

Philip Taffet: Sorry, Which channel?

Alexander Meißner: Virtual Machine

Philip Taffet: Virtual machine? Got it! Thank you!

Jacob Creech: Okay. Is there any other questions on the runtime. V: 2 and apologies. Mango team. I don't think we're going to be able to get tier. Your SIMD.Okay, cool.

So thank you all for coming to talk about this for the future agenda items.
What we're gonna do is we're gonna have it all on Github and I'm just gonna create a new issue for the next call, and if you want to propose a new item for the agenda, you can just leave it there. And then hopefully, we can get more information there as well as like documentation ahead of time, so that we can have a more productive conversation when we get to this call.

I think we might have to make this call longer in the future depending on what the agenda items are or do it more often. I'll reach out to you all to decide on how we do that.
But yeah, thank you. Everyone for coming today. Bye!