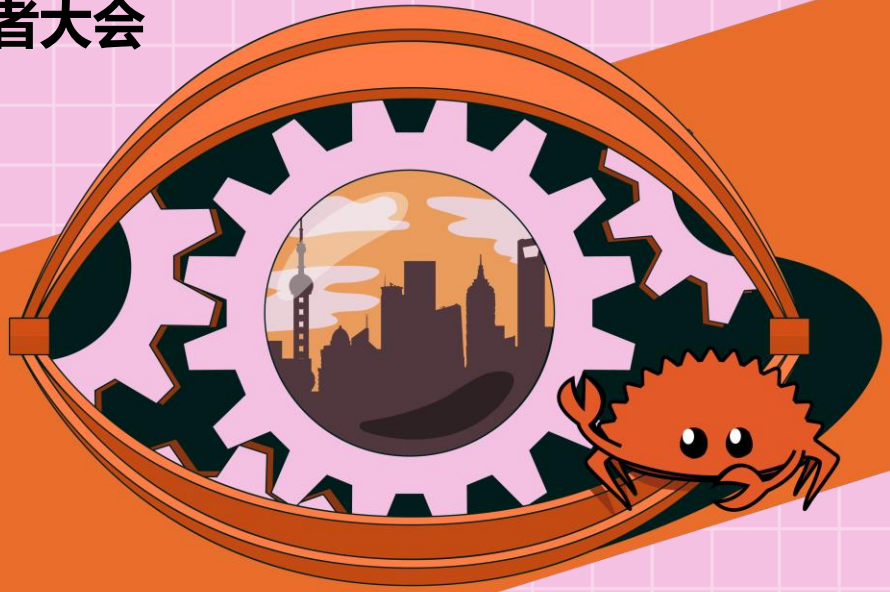
RUST CHINA CONF 2023

第三届中国Rust开发者大会



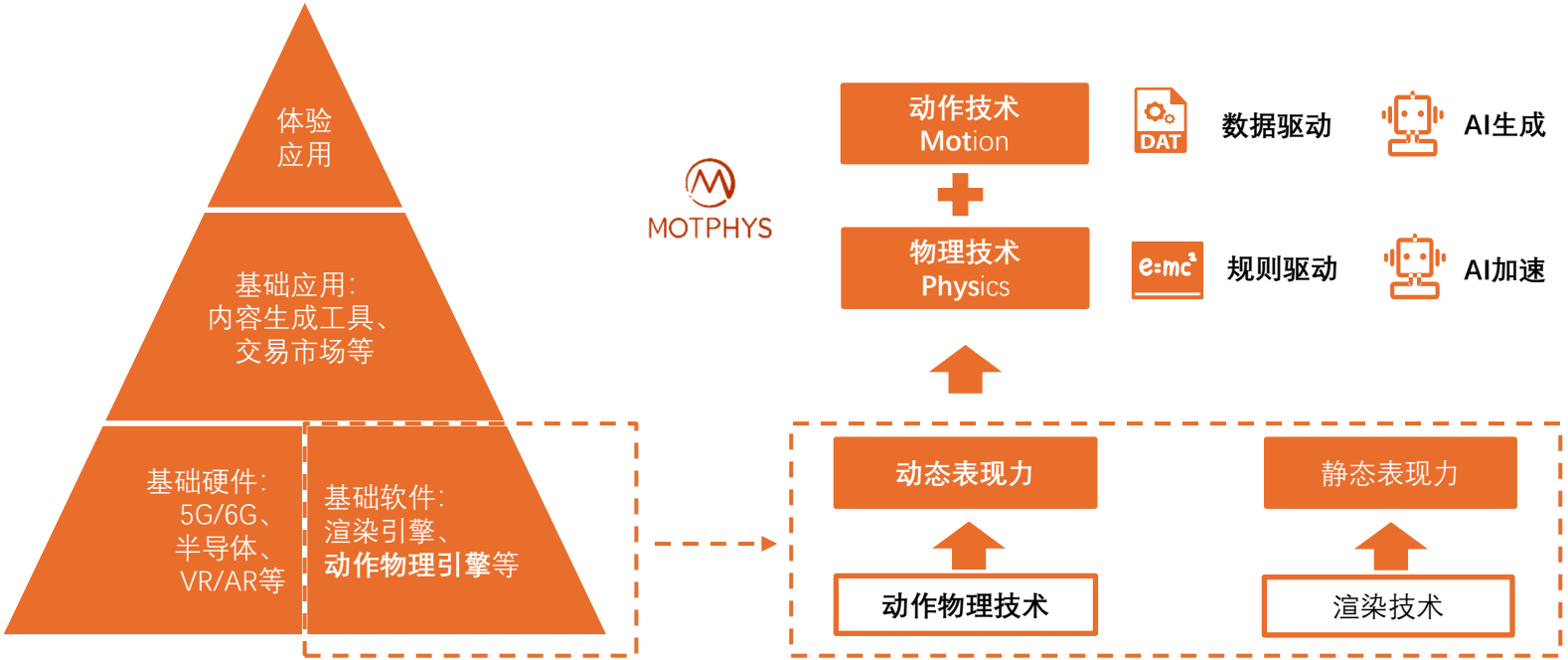
6.17-6.18 @Shanghai

Rust在物理引擎研发中的应用

崔汉青

Motphys CEO

Motphys 驱动虚拟世界的全部运动



云原生架构和 AI 能力

架构特点

性能特点

功能特点

MOTPHYS

云原生架构 | 分布式计算 | 跨平台确定性 | 算力动态调配 | 动作物理统一

AI仿真加速 | 算法优化 | 工程优化 | 物理材质统一解算

AI动作生成 | 前沿动作功能 | 优秀的易用性和适配性

PhysX by NVIDIA

havok

BULLET PHYSICS LIBRARY

Rapier

单机架构 | 动作物理分离

性能待优化 | 物理材质单独解算

缺乏AI能力

缺乏动作功能

Motphys 物理引擎的设计目标

保证每个目标平台的极致性能

跨端确定性 – 保证所有目标平台计算结果完全一致

具备分布式能力 – 通过横向扩展突破单机物理算力的上限

为什么选择 Rust

高性能

Rust 的性能和 C/C++ 比肩，支持 SIMD 优化，满足苛求性能的引擎研发需求； Rust 的零开销抽象甩掉了复杂设计的性能包袱

高表达力

Rust 在不损耗性能的情况下，其优秀的语法设计保证了语言的强大表达力：用更少的代码写更多的功能

安全

Rust 在语法层面极大程度保证了内存安全和并发安全

跨平台

Rust 依靠 LLVM 实现了多目标平台，并可以用语言内建的 target_feature 针对不同的指令集进行处理

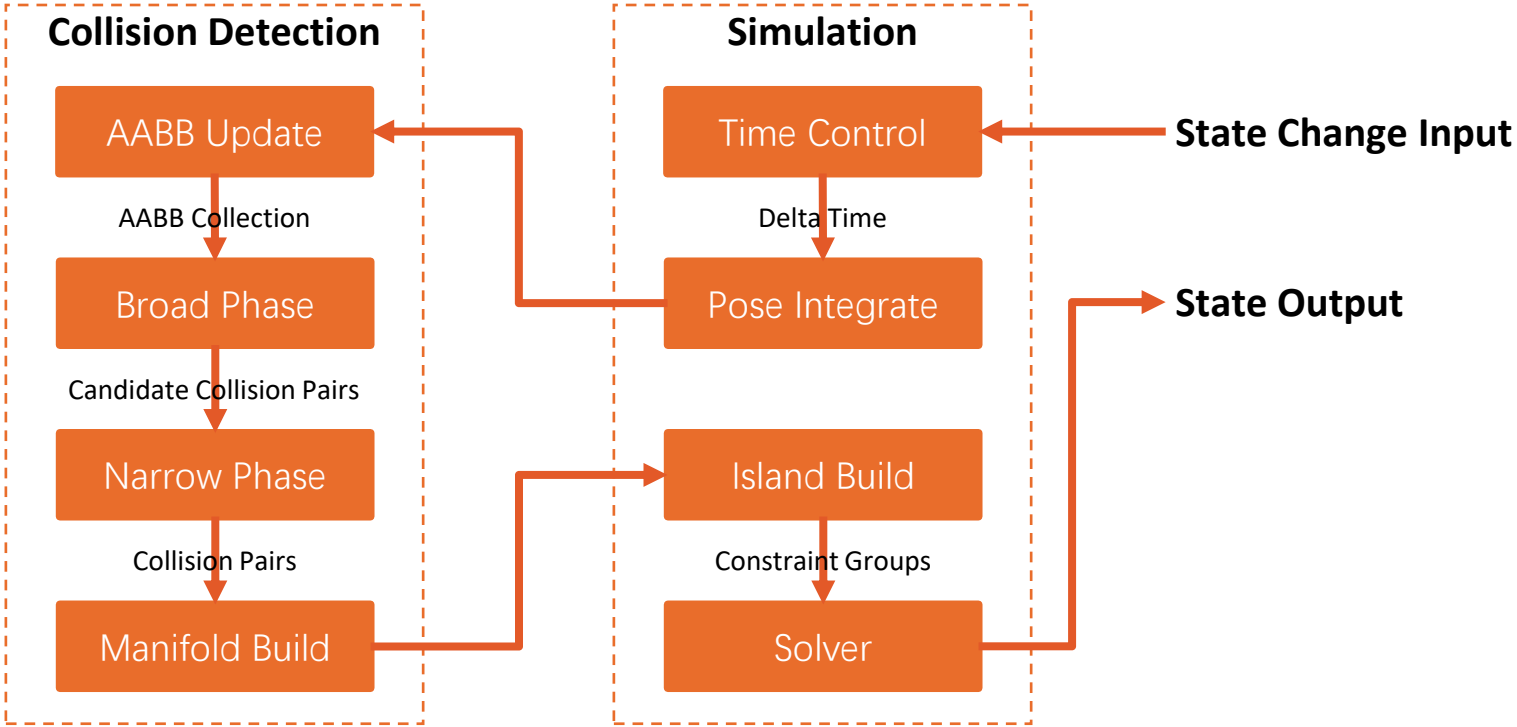
依赖管理

Cargo 真的比 cmake 好太多了

无惧并发

语言内建的 async/await，还有优秀的 crates rayon（计算密集型并发支持）和 tokio（IO 密集型并发支持）

Motphys 物理引擎架构



Rust 开源数学库的痛点

glam 代码质量高，使用简单，但是没有AoSoA类型，跨端确定性难以保证

nalgebra 过于复杂，大量的泛型导致使用不便，代码质量一般

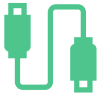
其余开源 crates 完成度不高

写好数学库并不容易

- 充分利用目标平台指令集 – 大量的针对目标平台的SIMD优化
- 数学计算本质上是类似的 – 大量的相似代码
- Portable SIMD unstable
- 影响跨端确定性的因素太多了

motphys-math

为高性能物理引擎量身定制



增加了 AoSoA 类型，并做了大量 SIMD 优化

增加了跨端确定性模式

保证所有设备浮点计算结果完全一致

性能超越目前开源的Rust基础数学库

glam

nalgebra

ultraviolet

Generic, procedure macro, or…

Generic

表达力不足

不容易做精细性能优化

Procedure Macro

过于复杂

结果不可见

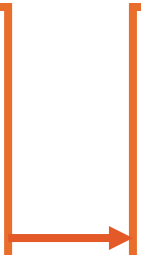
那么，用代码生成代码？

Web 开发用的模板引擎，
也可以用于生成 Rust 代码

tera

模板生成分指令集优化的 Rust 代码
提供远超 Procedure Macro 的可读性和易用性

```
#[inline]
pub(crate) fn dot(self, rhs: Self) -> {{ scalar_t }} {
    {% if is_ffi %}
        {{ non_ffi_self_t }}::dot(self.into(), rhs.into())
    {% else %}
        {% if is_unit %}
            self.{{ as_general_fn }}().dot(rhs.{{ as_general_fn }}())
        {% else %}
            {% if is_scalar %}
                {% for c in components %}
                    (self.{{ c }} * rhs.{{ c }}) {% if not loop.last %} + {% endif %}
                {%- endfor %}
            {% elif is_wasm32 %}
                crate::wasm32::dot{{ dim }}(self.0, rhs.0)
            {% elif use_simd %}
                unsafe { crate::{{ simd_instruction_set }}::dot{{ dim }}(self.0, rhs.0) }
            {% endif %}
        {% endif %}
    {% endif %}
}
```



```
#[inline]
pub(crate) fn dot(self, rhs: Self) -> f32 {
    crate::f32::simd_alias::Vec3A::dot(self.into(), rhs.into())
}

#[inline]
pub(crate) fn dot(self, rhs: Self) -> f32 {
    (self.x * rhs.x) + (self.y * rhs.y) + (self.z * rhs.z)
}
#[inline]
pub(crate) fn dot(self, rhs: Self) -> f32 {
    unsafe { crate::sse2::dot3(lhs: self.0, rhs: rhs.0) }
}
#[inline]
pub(crate) fn dot(self, rhs: Self) -> f32 {
    crate::wasm32::dot3(self.0, rhs.0)
}
```

mathbench

- 已有开源数学 crates 的 benchmark

- motphys-math benchmark 超越开源 crates

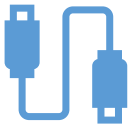
Benchmark	motphys-math	ultraviolet	nalgebra	ratio/nalgebra
matrix2 transpose x16	12.00 ns	32.92 ns	33.37 ns	2.8
matrix3 determinant x16	10.71 ns	12.01 ns	12.00 ns	1.1
matrix3 inverse x16	30.28 ns	39.53 ns	45.81 ns	1.5
transform point3 x16	18.42 ns	23.12 ns	20.91 ns	1.1

narrow phase benchmark

motphys-math 加持下的 narrow phase

Benchmark	motphys	PhysX 5	ratio
Capsule – Capsule – head to head penetrate	51 ns	219 ns	4.3
Capsule – Cuboid – coincide	166 ns	292 ns	1.8
Capsule – Infinite Plane – capsule one hemisphere penetrate	18 ns	63 ns	3.5
Sphere – Capsule – close at cylindrical part	12 ns	41 ns	3.4
Sphere – Cuboid – close at edge	5 ns	39 ns	7.8
Sphere – InfinitePlane – nocollide	9 ns	16 ns	1.8
Sphere – Sphere – coincide	7 ns	16 ns	2.3

Motphys 分布式物理引擎设计目标



0.02s内多次通信

苛刻的低延迟要求

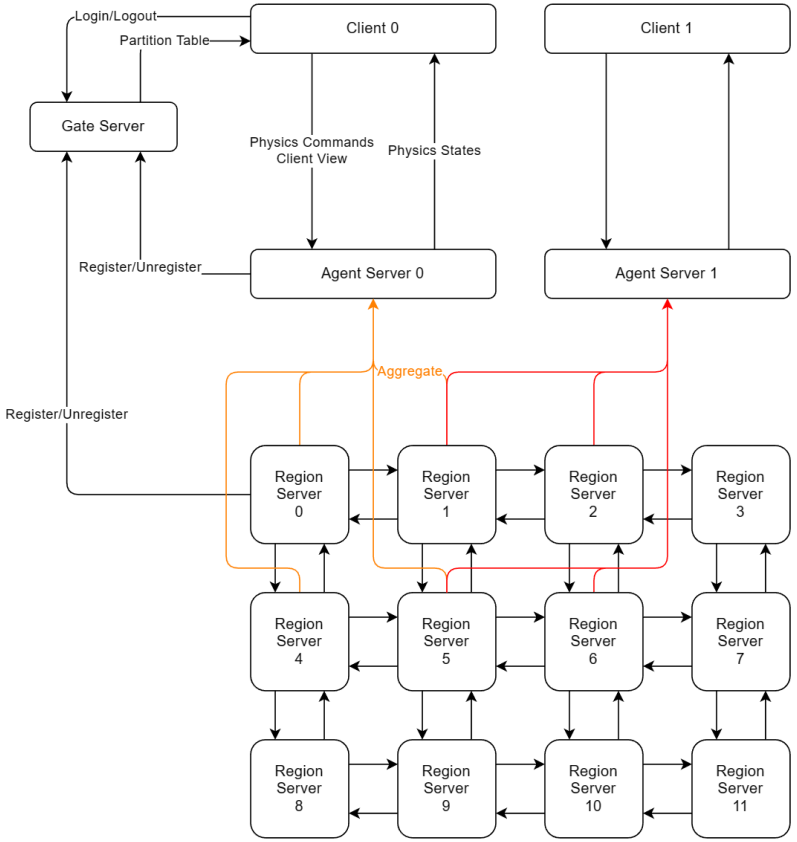
高速内网环境下的线性扩展 – 新增结点的网络开销恒定

高可用和负载均衡

Message的RTT

可测量

Motphys
分布式物理引擎网络架构



derive serde vs protobuf

- 通信协议选型
- protobuf/msgpack/…
  - 跨语言协议
  - 需要额外定义数据结构

- Pure Rust
  - 我们不需要跨语言
  - 复用已有的数据结构 Point/Vector…
  - derive serde
  - 可自定义serializer/deserializer，灵活度高
- Pure rust win

Pure Rust message

```rust
/// Region server p2p message
#[derive(Serialize, Deserialize, Debug)]
4 implementations
pub enum Message {
    NewPeer(NewPeer),
    NewPeerAccepted,
    BodyTransforms(BodyTransforms),
    BodyVelocities(BodyVelocities),
    ServerObjectMigrationOps(ServerObjectMigrationOps),
    BodyPreIntegrations(BodyPreIntegrations),
}

pub type BodyTransforms = BulkGBodyProperty<Isometry3>;
pub type BodyVelocities = BulkGBodyProperty<Velocity>;
pub type BodyPreIntegrations = BulkGBodyProperty<PreIntegrationData>;
```

```rust
/// A 3-dimensional vector.
#[derive(Clone, Copy, PartialEq)]
#[cfg_attr(not(target_arch = "spirv"), repr(C))]
#[cfg_attr(target_arch = "spirv", repr(simd))]
147 implementations
pub struct Vec3 {
    pub x: f32,
    pub y: f32,
    pub z: f32,
}

/// A 3D isometry transform, which can represent translation, rotation.
#[derive(Copy, Clone)]
#[repr(C)]
#[repr(align(16))]
47 implementations
pub struct Isometry3 {
    pub rotation: UnitQuat,
    pub translation: Vec3,
}
```

为了低延迟

- 序列化
  - 不考虑版本兼容性
  - 不带字段描述信息
  - 仅支持primitives, Vec<primitives>, 以及它们的组合嵌套

- 分布式物理引擎的计算和IO都很重要
  - 计算线程和IO线程分离，各自绑定CPU核心

Motphys 特化网络层

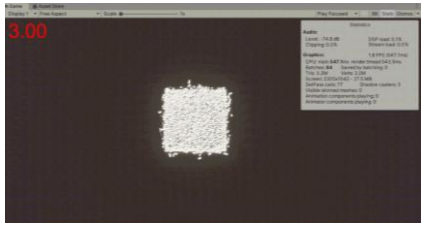
自定义 pure rust message

自定义编解码协议

自动 ack 和可测量 RTT 的通信框架

一切为低延迟服务

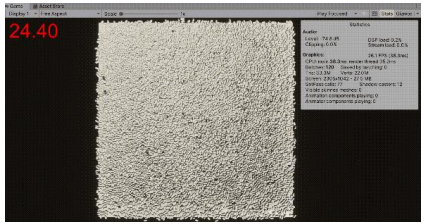
分布式物理，突破单机算力的瓶颈



Unity 测试案例

单机架构
同屏大规模物理量模拟
单机渲染帧率3fps

VS



Motphys 分布式架构

10倍于上述场景中的物理量
单机渲染帧率 25fps
物理集群帧率 50fps
此时物理模拟已不是算力瓶颈

横向扩展能力

单个节点的计算复杂度和网络通信复杂度，不会随集群总规模的上升而上升，**集群可线性扩容**

咪咕-星际广场：10万人级别同屏同步元宇宙项目

Unity + MOTPHYS

分布式渲染技术

分布式实时物理技术

全球首个 物理建模

基于云原生渲染和物理引擎的应用案例

全部用户之间、用户与道具和场景间均可实时物理交互

10万人

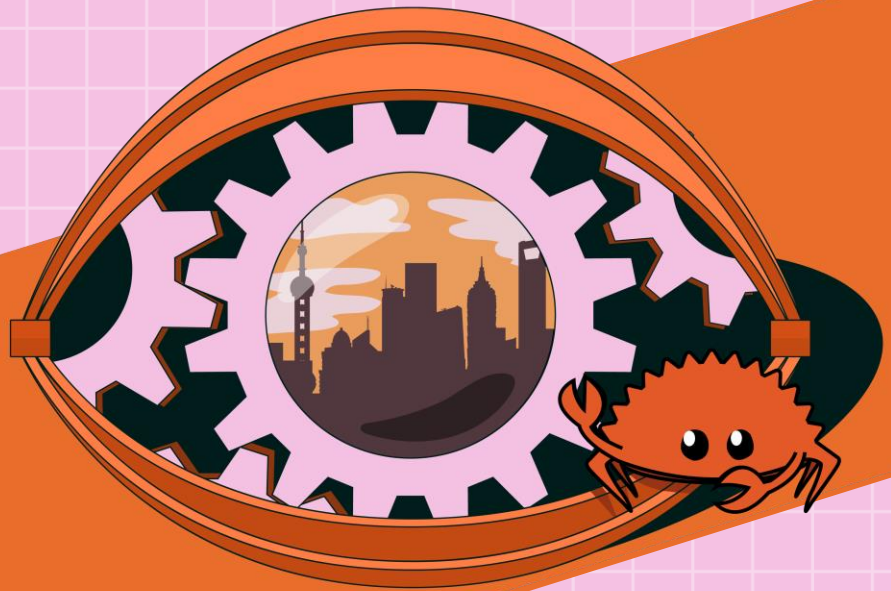
最高同时在线人数超过

12万平米

模拟场地面积达到

关于咪咕项目详细情况请参考：https://mp.weixin.qq.com/s/a_tvBU0wk9IwGYNqVK4EbA

Thank you!



Motphys