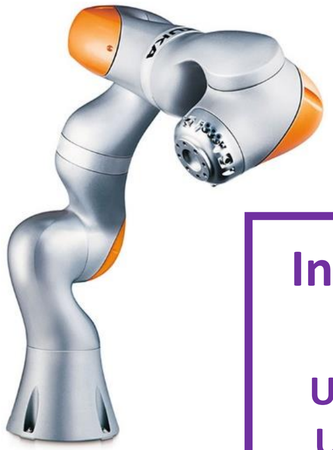
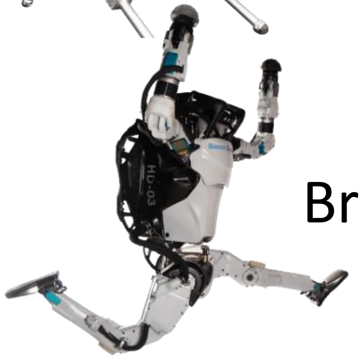
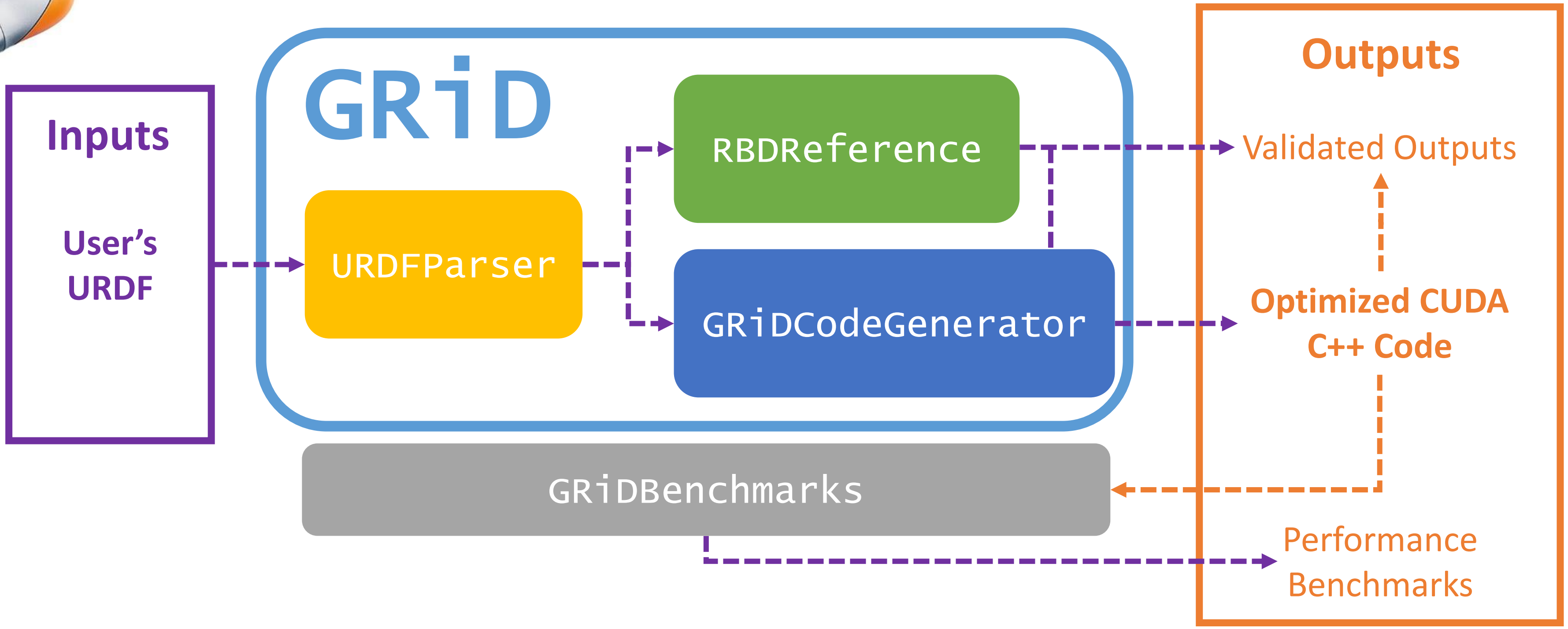
GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here







Brian Plancher[1], Sabrina M. Neuman[1], Radhika Ghosal[1], Scott Kuindersma[1,2], Vijay Janapa Reddi[1]

1: Harvard University John A. Paulson School of Engineering and Applied Sciences, 2: Boston Dynamics

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

GRiD makes it easy to use the GPU with robotics algorithms that use rigid body dynamics and provides up to a **7.2x speedup** and maintains **a 2.5x speedup with I/O**.

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

1. Why GPU Rigid Body Dynamics?

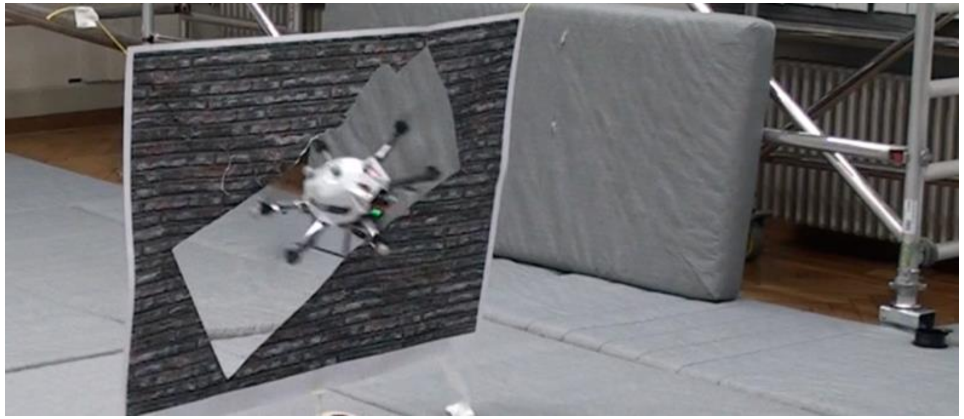
2. GRiD's Modular Design

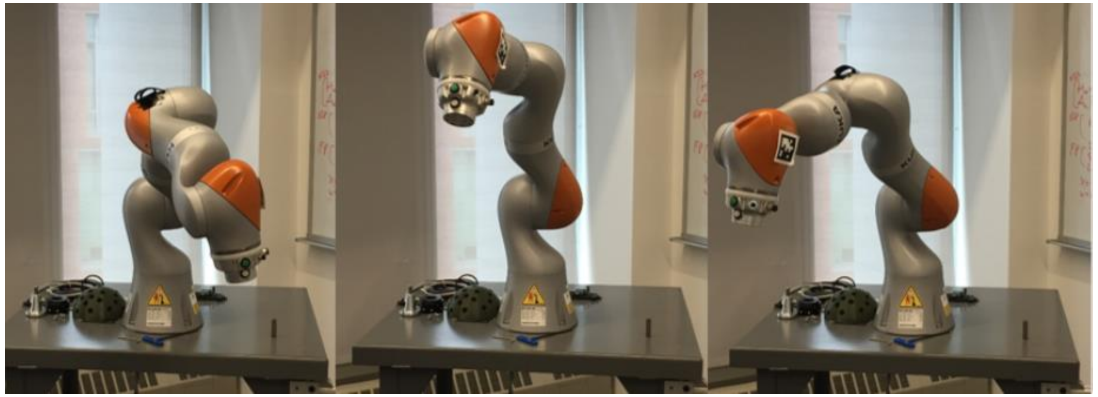
3. GRiD's Optimizations

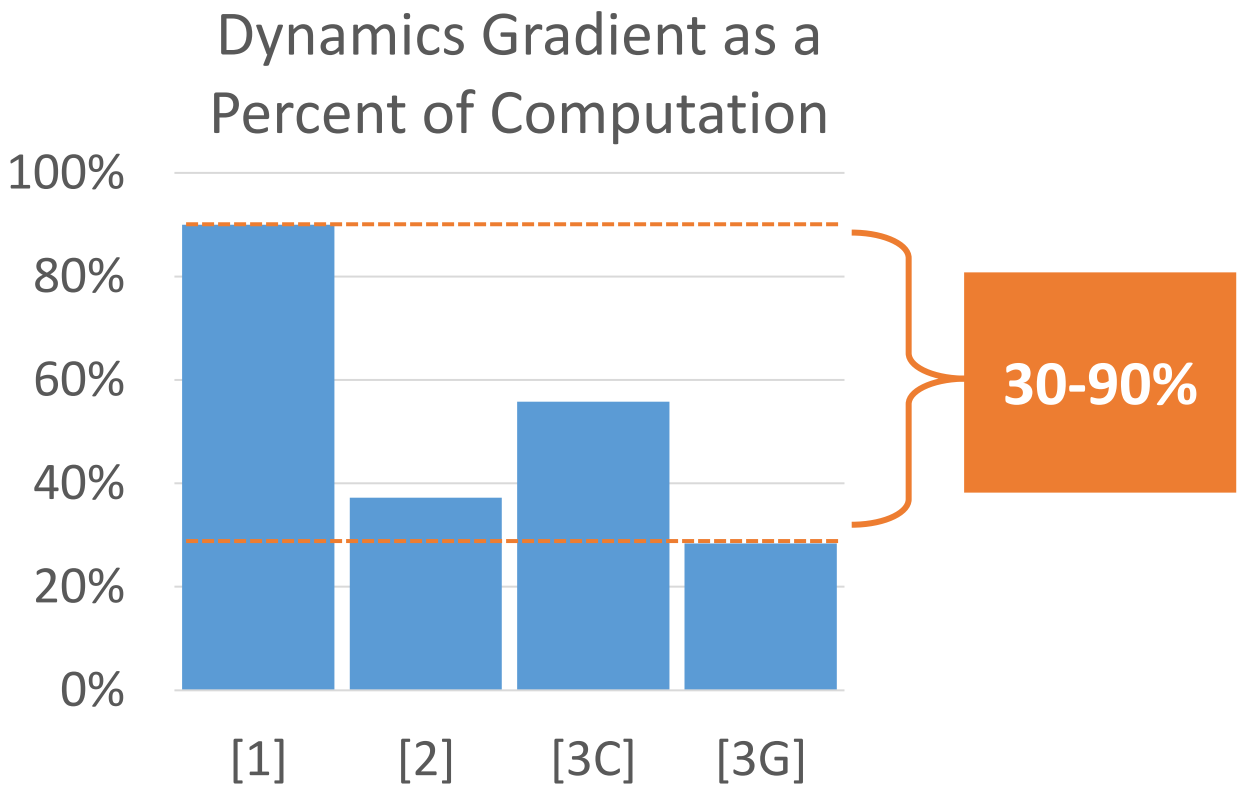
4. Results

Rigid Body Dynamics Gradients are a bottleneck for planning and control (e.g., nonlinear MPC)

Place Zoom Headshot Here







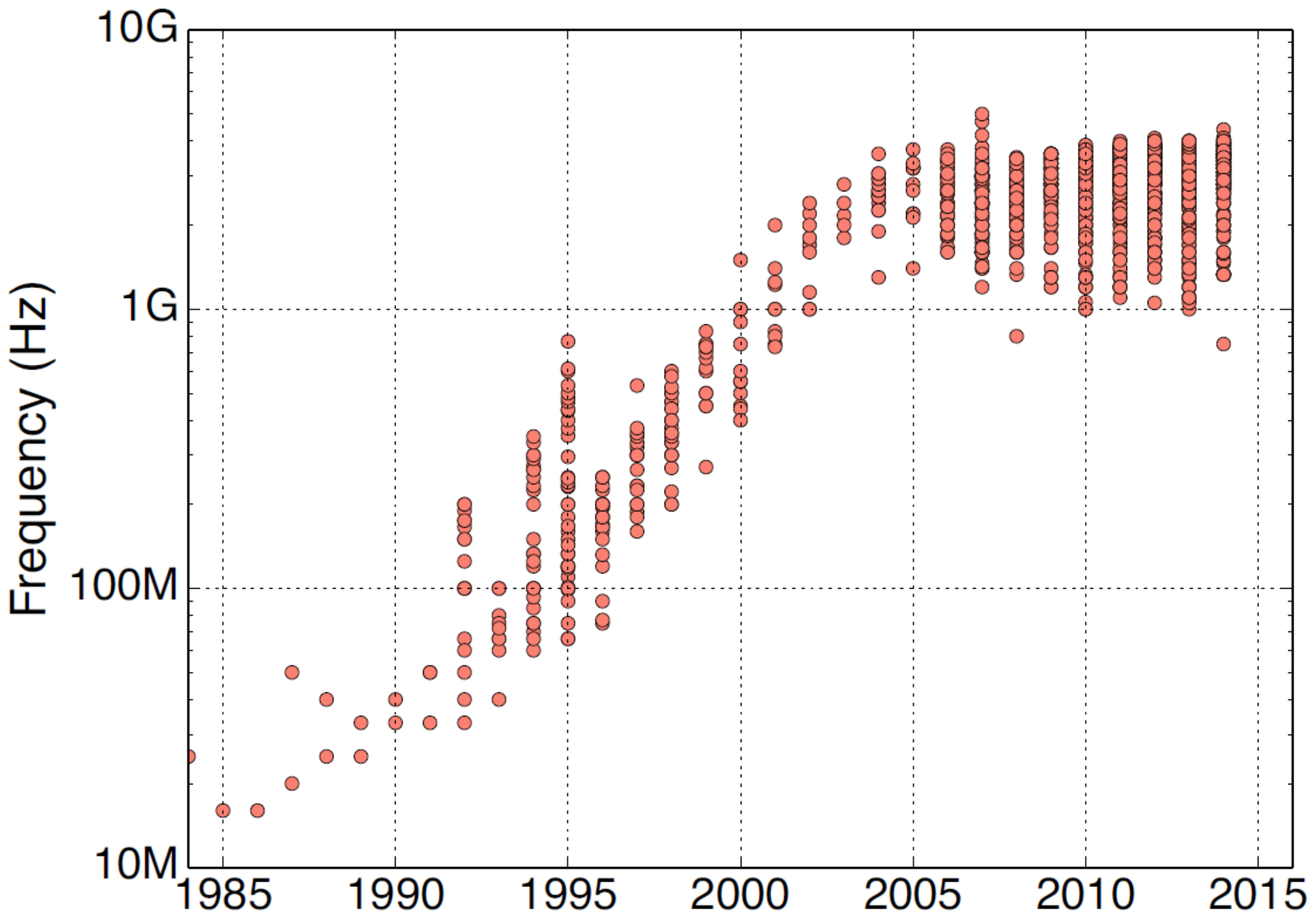
[1] J. Carpentier and N. Mansrud, "Analytical Derivatives of Rigid Body Dynamics Algorithms," RSS 2018

[2] M. Neunert, et al., "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking," ICRA 2016

[3] Best end-to-end [C]PU and [G]PU option from B. Plancher and S. Kuindersma, "A Performance Analysis of Parallel Differential Dynamic Programming," WAFR 2018

CPUs aren't getting faster

Place Zoom Headshot Here



- Frequency scaling is ending (CPUs aren't getting faster)
- Massive parallelism on GPUs may be a solution for hardware acceleration

[Shao and Brooks "Synthesis Lectures on Computer Architecture" 2015]

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

1. Why GPU Rigid Body Dynamics?

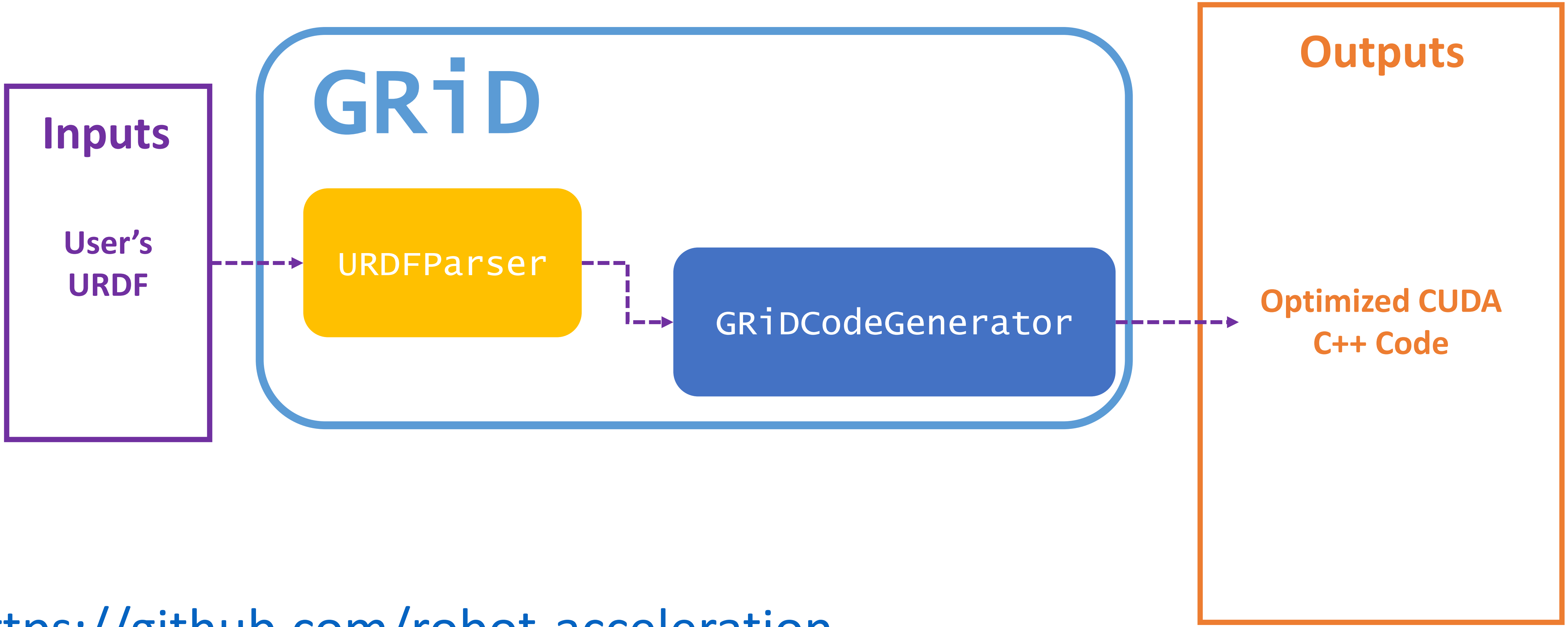
2. GRiD's Modular Design

3. GRiD's Optimizations

4. Results

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

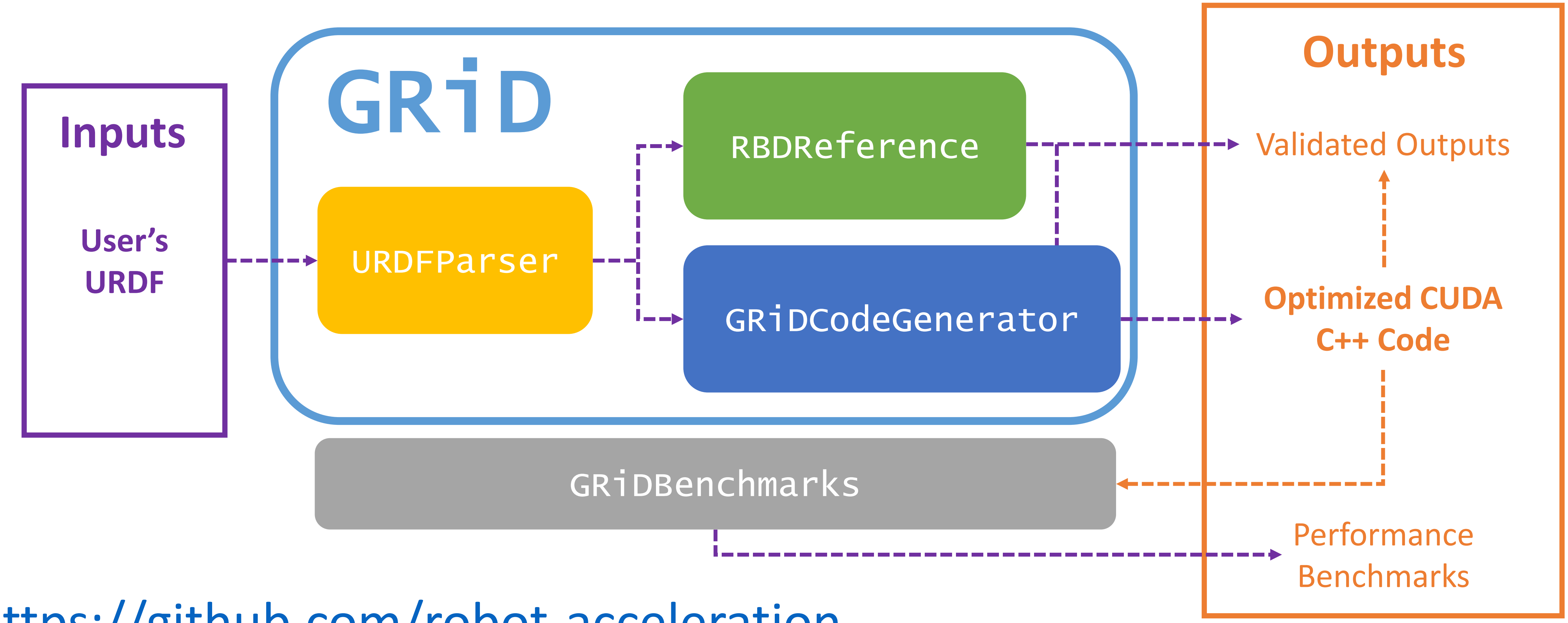
Place Zoom Headshot Here



https://github.com/robot-acceleration

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here



https://github.com/robot-acceleration

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

https://github.com/robot-acceleration

GRiD currently supports:

- Prismatic, fixed, and revolute joints
- ID, FD, M^{-1}
- ∇ID, ∇FD with respect to $\mathbf{q}, \dot{q}, \mathbf{u}$

We are actively working to expand these features and welcome community support in this effort!

GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

1. Why GPU Rigid Body Dynamics?

2. GRiD’s Modular Design

3. **GRiD’s Optimizations**

4. Results

GRiD exploits the structure of each robot to minimize memory and optimize latency

Place Zoom Headshot Here

Very serial algorithm

Algorithm 1 ∇RNEA-F$(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

for frame $i = 1 : N$ **do**

$$\frac{\partial v_i}{\partial u} = {}^iX_{\lambda_i}\frac{\partial v_{\lambda_i}}{\partial u} + \begin{cases} \left({}^iX_{\lambda_i}v_{\lambda_i}\right) \times S_i & u \equiv q \\ S_i & u \equiv \dot{q} \end{cases}$$

3: $$\frac{\partial a_i}{\partial u} = {}^iX_{\lambda_i}\frac{\partial a_{\lambda_i}}{\partial u} + \frac{\partial v_{\lambda_i}}{\partial u} \times S_i\dot{q}_i + \begin{cases} \left({}^iX_{\lambda_i}a_{\lambda_i}\right) \times S_i \\ v_i \times S_i \end{cases}$$

4: $$\frac{\partial f_i}{\partial u} = I_i\frac{\partial a_i}{\partial u} + \frac{\partial v_i}{\partial u} \times^* I_i v_i + v_i \times^* I_i\frac{\partial v_i}{\partial u}$$

GRiD exploits the structure of each robot to minimize memory and optimize latency

Place Zoom Headshot Here

Algorithm 2 ∇RNEA-F-GRiD$(\dot{q}, v, a, f, X, S, I) \rightarrow \partial f / \partial u$

for frame $i = 1 : n$ **in parallel do**

$\quad \alpha_i = {}^{i}X_{\lambda_i} v_{\lambda_i} \quad \beta_i = {}^{i}X_{\lambda_i} a_{\lambda_i} \quad \gamma_i = I_i v_i$

$\quad \alpha_i = \alpha_i \times S_i \quad \beta_i = \beta_i \times S_i \quad \delta_i = v_i \times S_i$

for level $l = 0 : l_{max}$ **do**

$\quad$ **for** frame $i \in l$ **in parallel do**

$$\frac{\partial v_i}{\partial u} = {}^{i}X_{\lambda_i} \frac{\partial v_{\lambda_i}}{\partial u} + \begin{cases} \alpha_i & u \equiv q \\ S_i & u \equiv \dot{q} \end{cases}$$

for frame $i = 1 : n$ **in parallel do**

$$\rho_i = \frac{\partial v_{\lambda_i}}{\partial u} \times S_i \dot{q}_i + \begin{cases} \beta_i \\ \delta_i \end{cases}$$

for level $l = 0 : l_{max}$ **do**

$\quad$ **for** frame $i \in l$ **in parallel do**

$$\frac{\partial a_i}{\partial u} = {}^{i}X_{\lambda_i} \frac{\partial a_{\lambda_i}}{\partial u} + \rho_i$$

for frame $i = 1 : n$ **in parallel do**

$$\frac{\partial f_i}{\partial u} = \frac{\partial v_i}{\partial u} \times^* \gamma_i \qquad \eta_i = v_i \times^* I_i$$

$$\frac{\partial f_i}{\partial u} = \frac{\partial f_i}{\partial u} + I_i \frac{\partial a_i}{\partial u} + \eta_i \frac{\partial v_i}{\partial u}$$

Refactor algorithms to expose **parallel loops of unified operations**

GRiD exploits the structure of each robot to minimize memory and optimize latency

Place Zoom Headshot Here

Algorithm 2 ∇RNEA-F-GRiD$(\dot{q}, v, a, f, X, S, I) \rightarrow \partial f / \partial u$

for frame $i = 1 : n$ **in parallel do**

$\alpha_i = {}^iX_{\lambda_i} v_{\lambda_i} \quad \beta_i = {}^iX_{\lambda_i} a_{\lambda_i} \quad \gamma_i = I_i v_i$

$\alpha_i = \alpha_i \times S_i \quad \beta_i = \beta_i \times S_i \quad \delta_i = v_i \times S_i$

for level $l = 0 : l_{max}$ **do**

for frame $i \in l$ **in parallel do**

$$\frac{\partial v_i}{\partial u} = {}^iX_{\lambda_i} \frac{\partial v_{\lambda_i}}{\partial u} + \begin{cases} \alpha_i & u \equiv q \\ S_i & u \equiv \dot{q} \end{cases}$$

for frame $i = 1 : n$ **in parallel do**

$$\rho_i = \frac{\partial v_{\lambda_i}}{\partial u} \times S_i \dot{q}_i + \begin{cases} \beta_i \\ \delta_i \end{cases}$$

for level $l = 0 : l_{max}$ **do**

for frame $i \in l$ **in parallel do**

$$\frac{\partial a_i}{\partial u} = {}^iX_{\lambda_i} \frac{\partial a_{\lambda_i}}{\partial u} + \rho_i$$

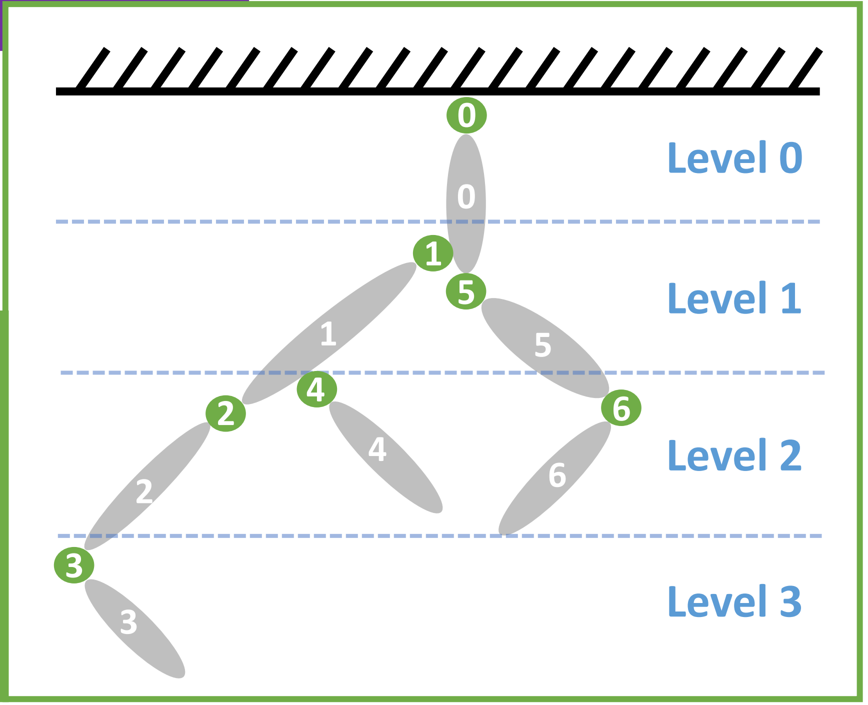
for frame $i = 1 : n$ **in parallel do**

$\frac{\partial f_i}{\partial u} = \frac{\partial v_i}{\partial u} \times^* \gamma_i \quad \eta_i = v_i \times^* I_i$

$\frac{\partial f_i}{\partial u} = \frac{\partial f_i}{\partial u} + I_i \frac{\partial a_i}{\partial u} + \eta_i \frac{\partial v_i}{\partial u}$

Refactor algorithms to expose **parallel loops of unified operations**

Compute remaining serial operations in **parallel across levels** of the rigid body tree



GRiD exploits the structure of each robot to minimize memory and optimize latency

Place Zoom Headshot Here

Algorithm 2 ∇RNEA-F-GRiD$(\dot{q}, v, a, f, X, S, I) \rightarrow \partial f / \partial u$

for frame $i = 1 : n$ **in parallel do**

$\quad \alpha_i = {}^iX_{\lambda_i} v_{\lambda_i} \quad \beta_i = {}^iX_{\lambda_i} a_{\lambda_i} \quad \gamma_i = I_i v_i$

$\quad \alpha_i = \alpha_i \times S_i \quad \beta_i = \beta_i \times S_i \quad \delta_i = v_i \times S_i$

for level $l = 0 : l_{max}$ **do**

$\quad$ **for** frame $i \in l$ **in parallel do**

$$\frac{\partial v_i}{\partial u} = {}^iX_{\lambda_i} \frac{\partial v_{\lambda_i}}{\partial u} + \begin{cases} \alpha_i & u \equiv q \\ S_i & u \equiv \dot{q} \end{cases}$$

for frame $i = 1 : n$ **in parallel do**

$$\rho_i = \frac{\partial v_{\lambda_i}}{\partial u} \times S_i \dot{q}_i + \begin{cases} \beta_i \\ \delta_i \end{cases}$$

for level $l = 0 : l_{max}$ **do**

$\quad$ **for** frame $i \in l$ **in parallel do**

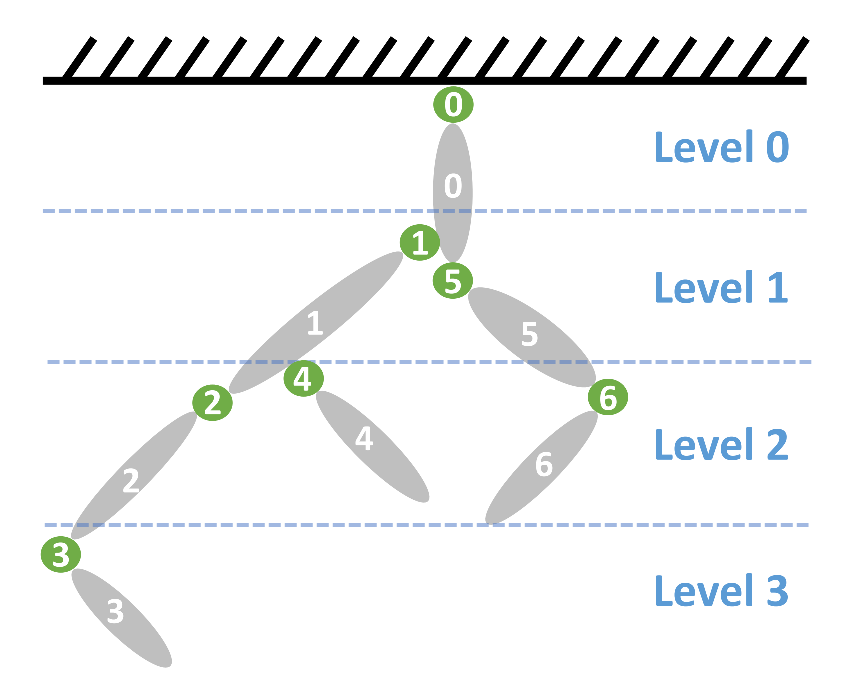
$$\frac{\partial a_i}{\partial u} = {}^iX_{\lambda_i} \frac{\partial a_{\lambda_i}}{\partial u} + \rho_i$$

for frame $i = 1 : n$ **in parallel do**

$$\frac{\partial f_i}{\partial u} = \frac{\partial v_i}{\partial u} \times^* \gamma_i \qquad \eta_i = v_i \times^* I_i$$

$$\frac{\partial f_i}{\partial u} = \frac{\partial f_i}{\partial u} + I_i \frac{\partial a_i}{\partial u} + \eta_i \frac{\partial v_i}{\partial u}$$

The **branch structure** also determines **sparsity in the columns of ∂v, ∂a, and ∂f**



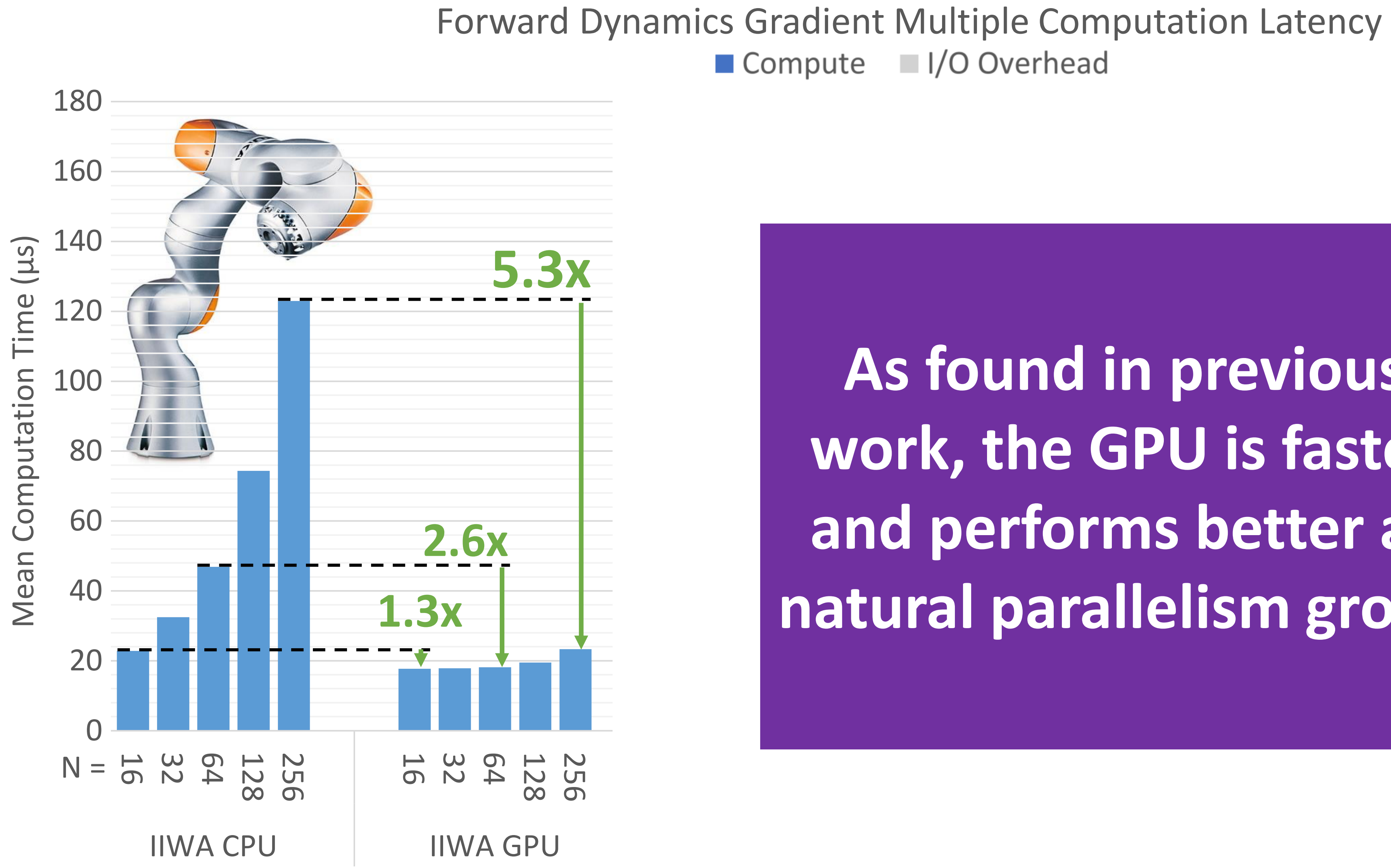
GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

1. Why GPU Rigid Body Dynamics?
2. GRiD's Modular Design
3. GRiD's Optimizations
4. **Results**

GRiD improves both computational latency and scalability

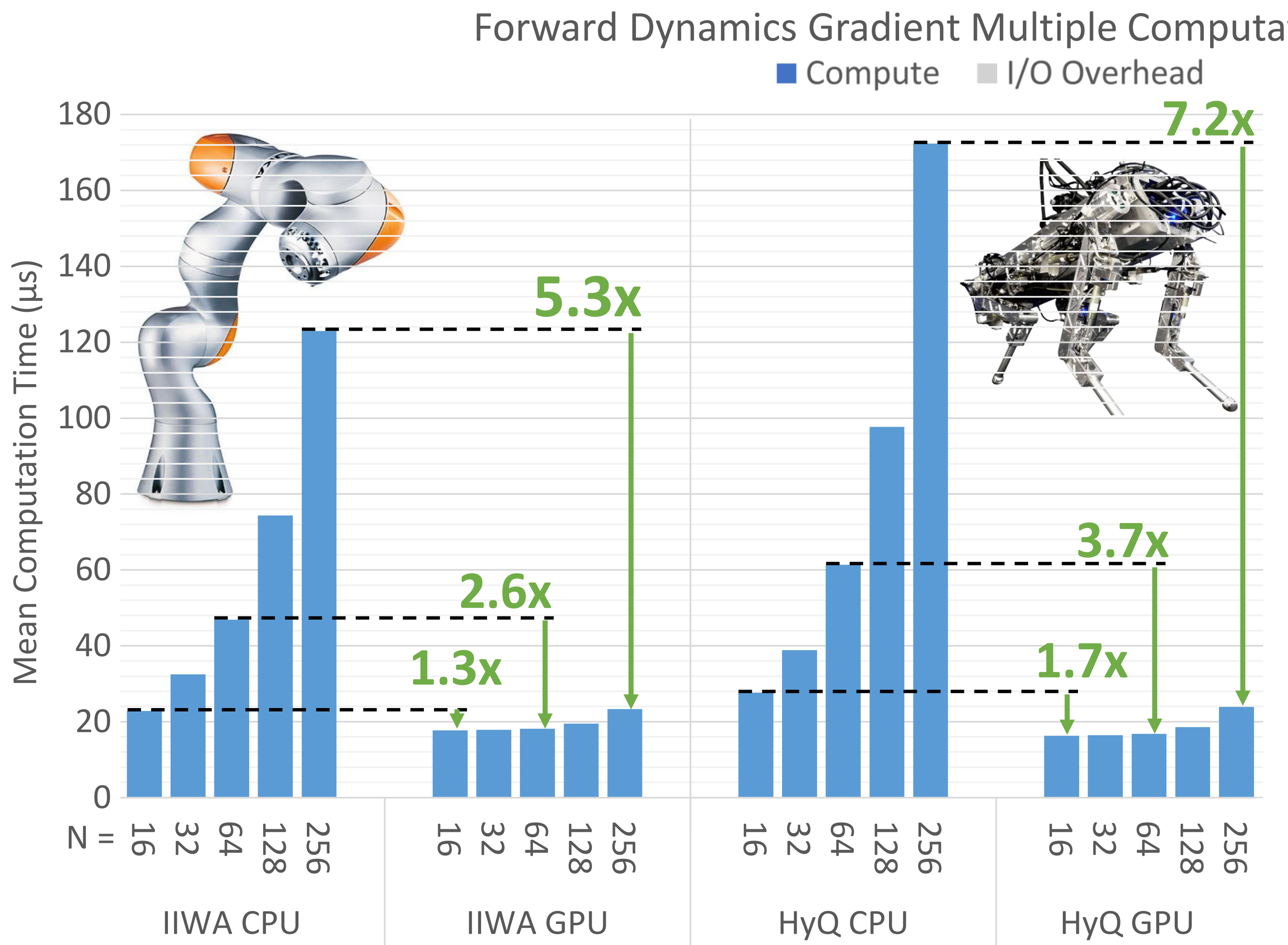
Place Zoom Headshot Here



As found in previous work, the GPU is faster and performs better as natural parallelism grows

GRiD improves both computational latency and scalability

Place Zoom Headshot Here

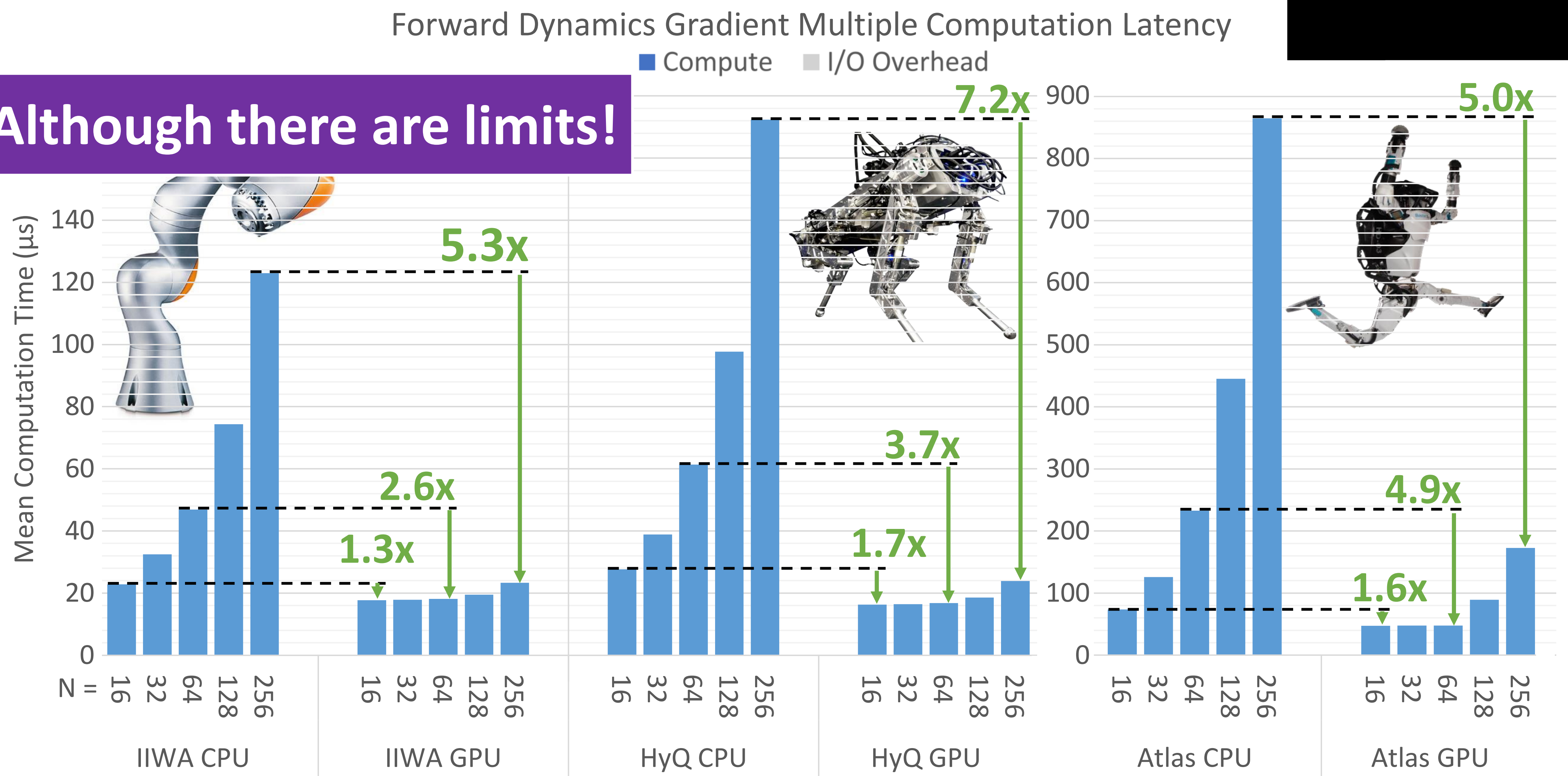


We show that the GPU does even better as robot complexity grows as well!

GRiD improves both computational latency and scalability

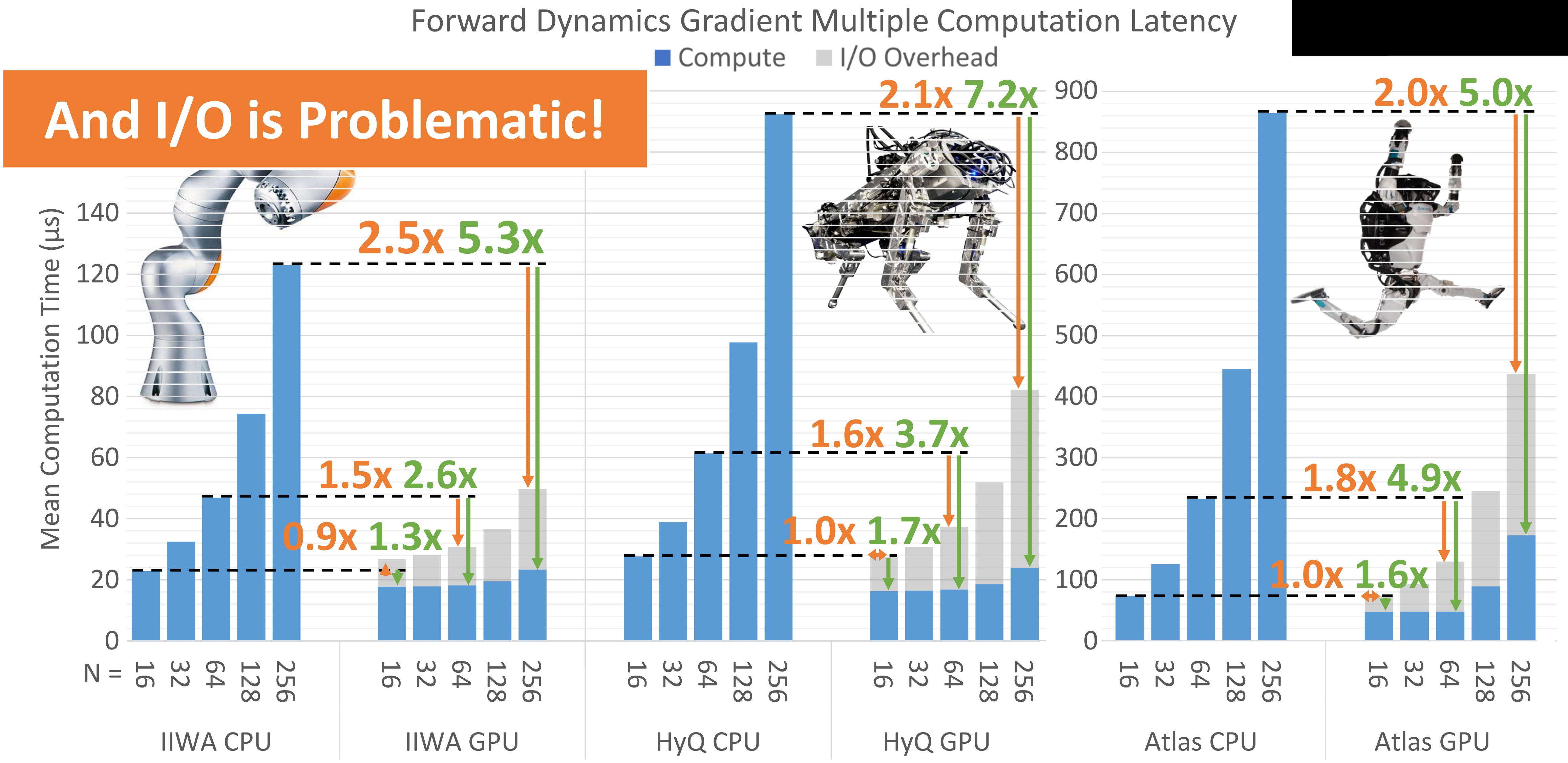
Place Zoom Headshot Here

Although there are limits!



GRiD improves both computational latency and scalability

Place Zoom Headshot Here



GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients

Place Zoom Headshot Here

GRiD is a **URDF to optimized CUDA C++ library** designed to provide **GPU acceleration** for rigid body dynamics algorithms and their analytical gradients. GRiD provides up to a **7.2x speedup** and maintains **a 2.5x speedup with I/O**.

GRiD makes it easy to use the GPU with robotics algorithms that use rigid body dynamics!

https://github.com/robot-acceleration

brian_plancher@g.harvard.edu

Harvard John A. Paulson School of Engineering and Applied Sciences

This material is based upon work supported by the National Science Foundation (under Grant DGE1745303 and Grant 2030859 to the Computing Research Association for the CIFellows Project). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organizations