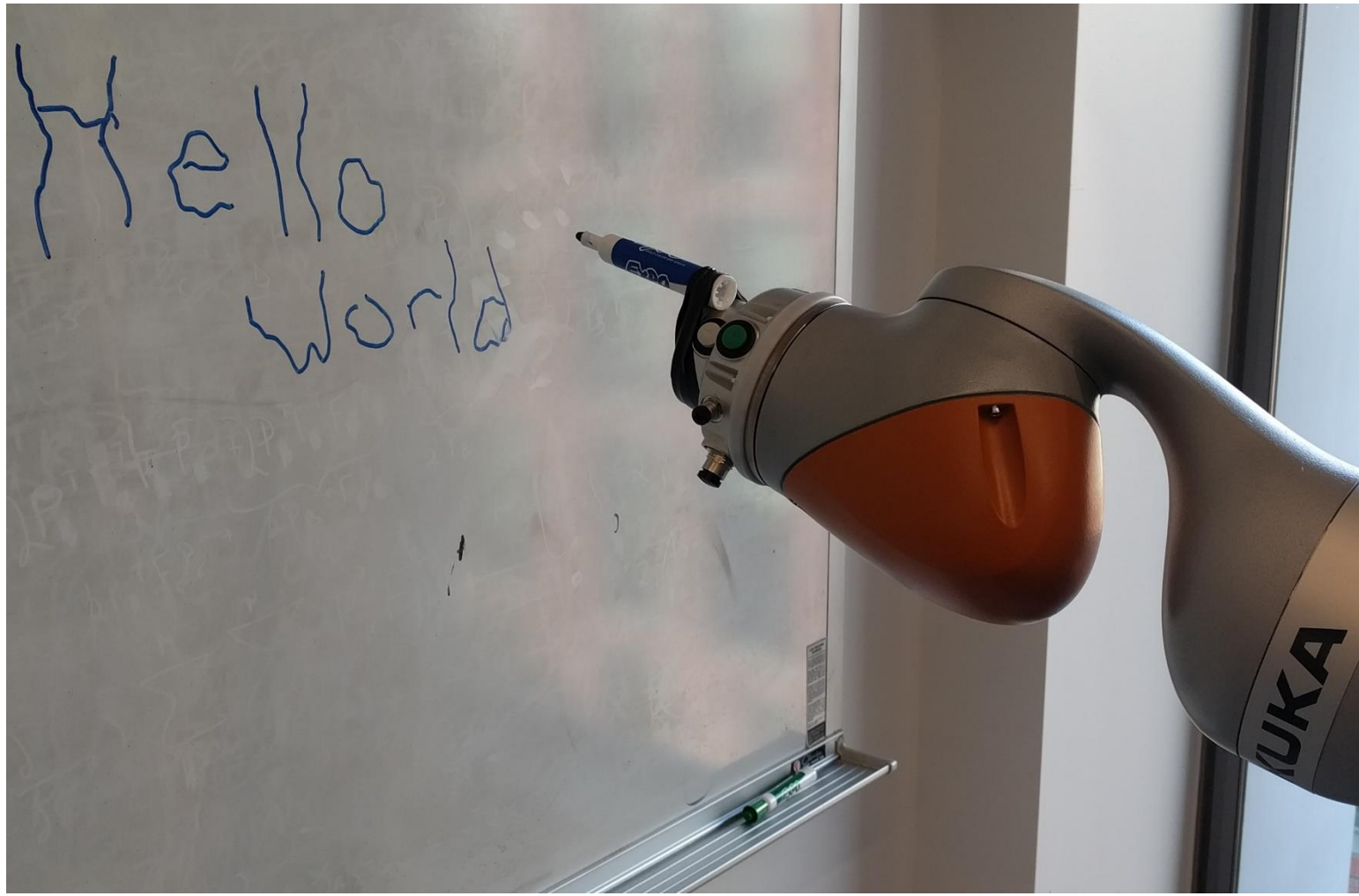
Hardware Acceleration for Realtime Robotics

Brian Plancher
12/14/21

Hi I’m Brian!



Hi I’m Brian!

- **I’m obsessed with my dog Alvin**

Hi I’m Brian!

- I’m obsessed with my dog Alvin **and my daughter Tess**

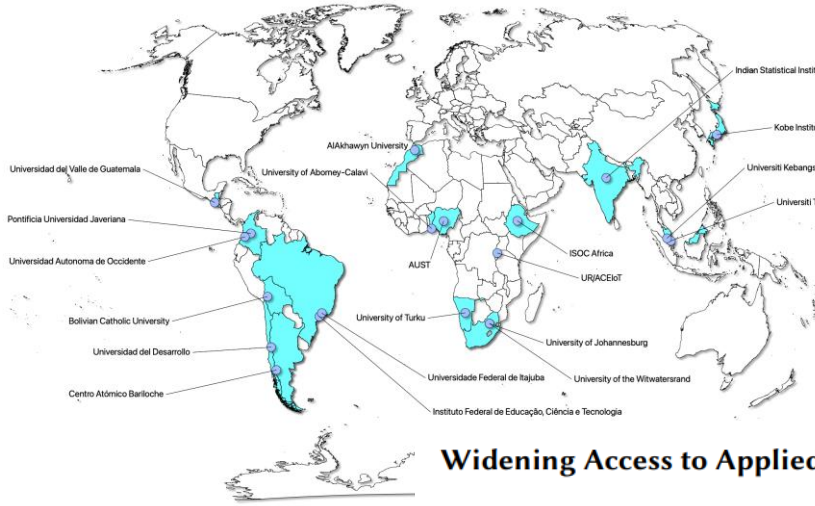
Hi I’m Brian!

- I’m obsessed with my dog Alvin and my daughter Tess
- **I am passionate about teaching accessible, interdisciplinary hands-on, project-based courses**

Hi I'm Brian!

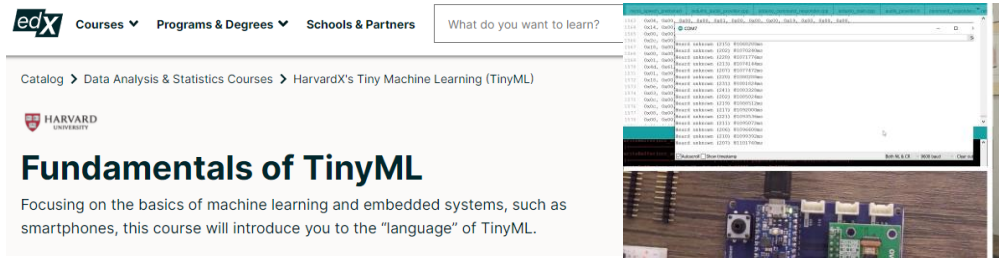
- I'm obsessed with my dog Alvin and my daughter Tess
- **I am passionate about teaching accessible, interdisciplinary hands-on, project-based courses**

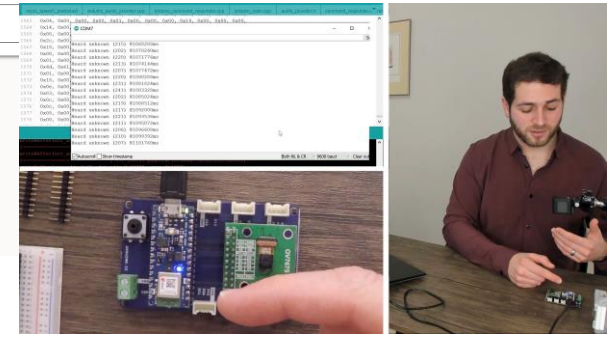
https://TinyMLedu.org



Widening Access to Applied Machine Learning with TinyML

Vijay Janapa Reddi* Brian Plancher* Susan Kennedy* Laurence Moroney†
Pete Warden† Anant Agarwal*‡ Colby Banbury* Massimo Banzi§ Matthew Bennett*
Benjamin Brown* Sharad Chitlangia¶ Radhika Ghosal* Sarah Grafman* Rupert Jaeger‖
Srivatsan Krishnan* Maximilian Lam* Daniel Leiker‖ Cara Mann* Mark Mazumder*
Dominic Pajak§ Dhilan Ramaprasad* J. Evan Smith* Matthew Stewart* Dustin Tingley*





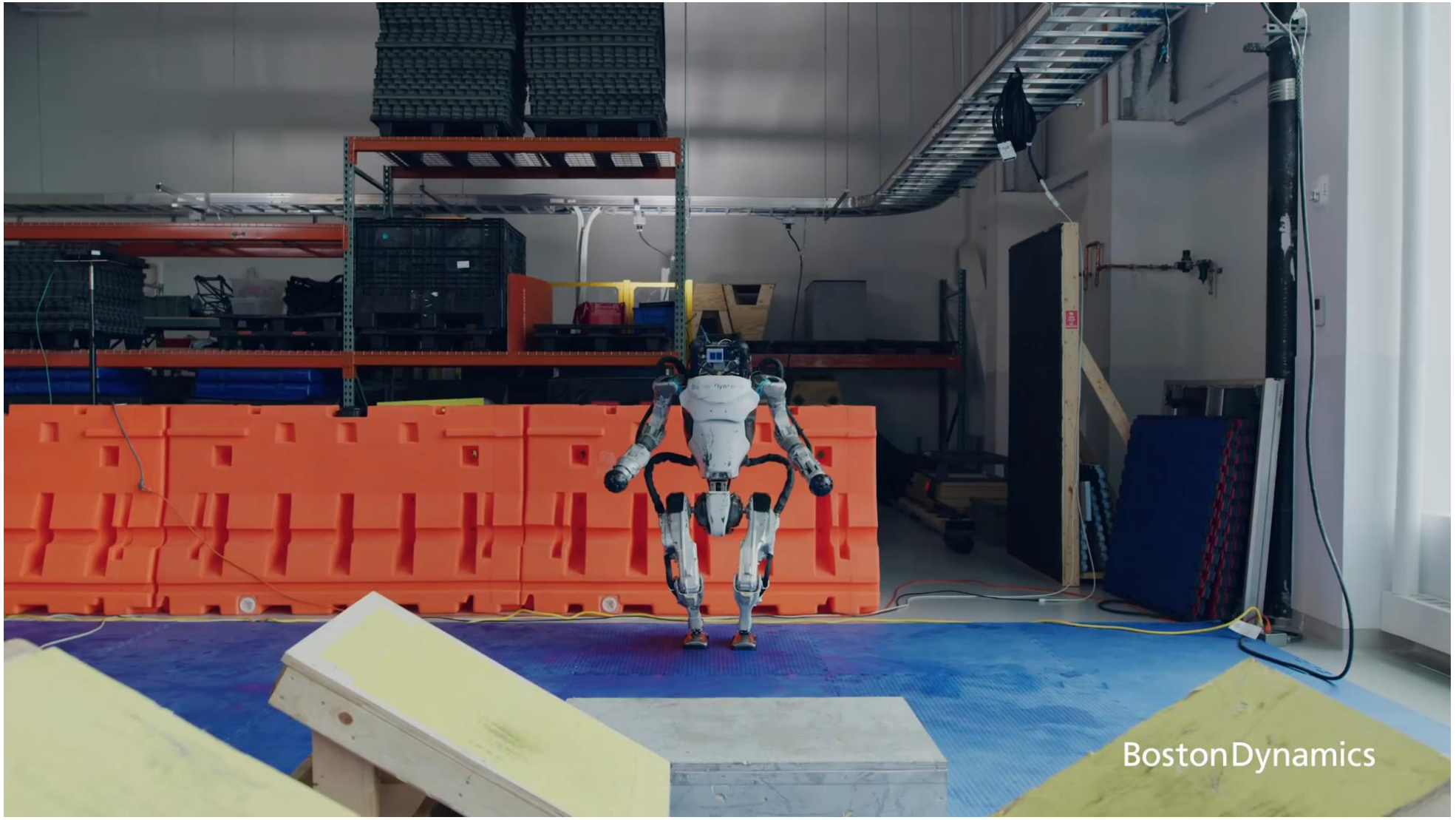
Hi I'm Brian!

- I'm obsessed with my dog Alvin and my daughter Tess
- I am passionate about teaching accessible, interdisciplinary hands-on, project-based courses
- **My research focuses on developing open-source planning and control algorithms that enable robots to operate in the real world and help people!**

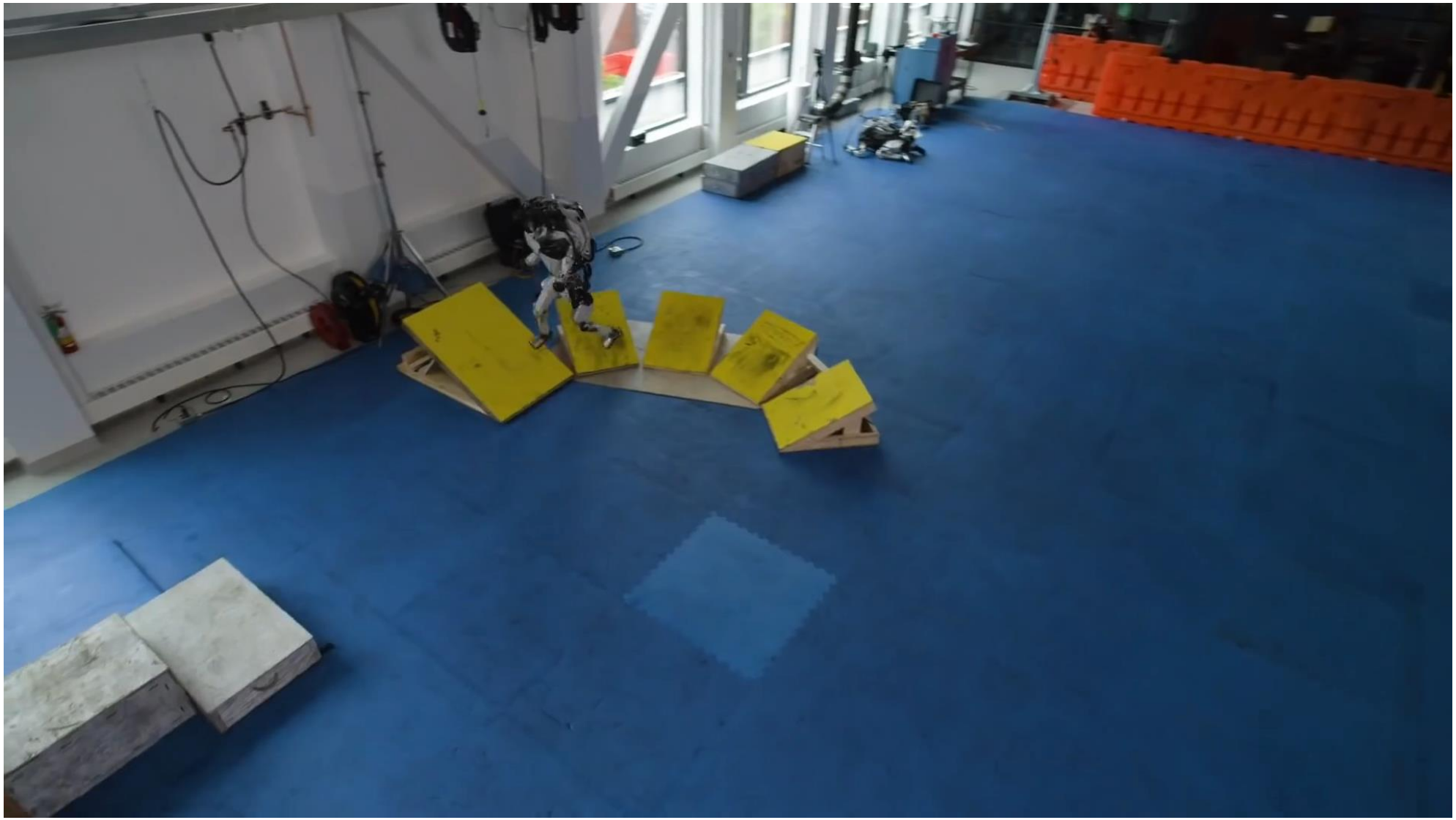
How about you?

1. How many people in the audience have a degree in computer science? (e.g., Faculty, Staff)
2. How many people have taken a handful of CS courses? (e.g., Junior or Senior CS Majors)
3. How about one or two CS courses?
4. How many people have no CS background?

Robots can do amazing things …



… but they still have a long way to go!



How can we fix this?

How can we fix this?

Be Smarter = Better Algorithms

Think Faster = Hardware Acceleration

Learning Goals for Today

1. Learn some of the **language of robotics**
   (and computer architecture)
2. Understand the important of **parallelism**
3. Gain practice in exploring the **process** and **opportunities** of **hardware acceleration**

Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

The Future of Accelerating Robotics

Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

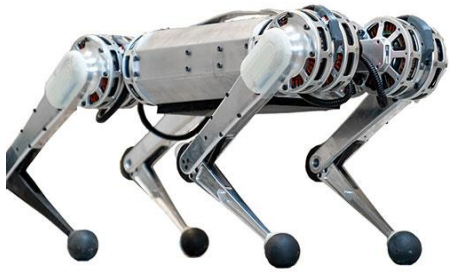
A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

The Future of Accelerating Robotics

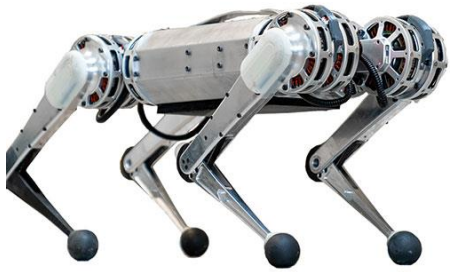
Robotics is a **BIG** space

Sensors → Compute → Actuators

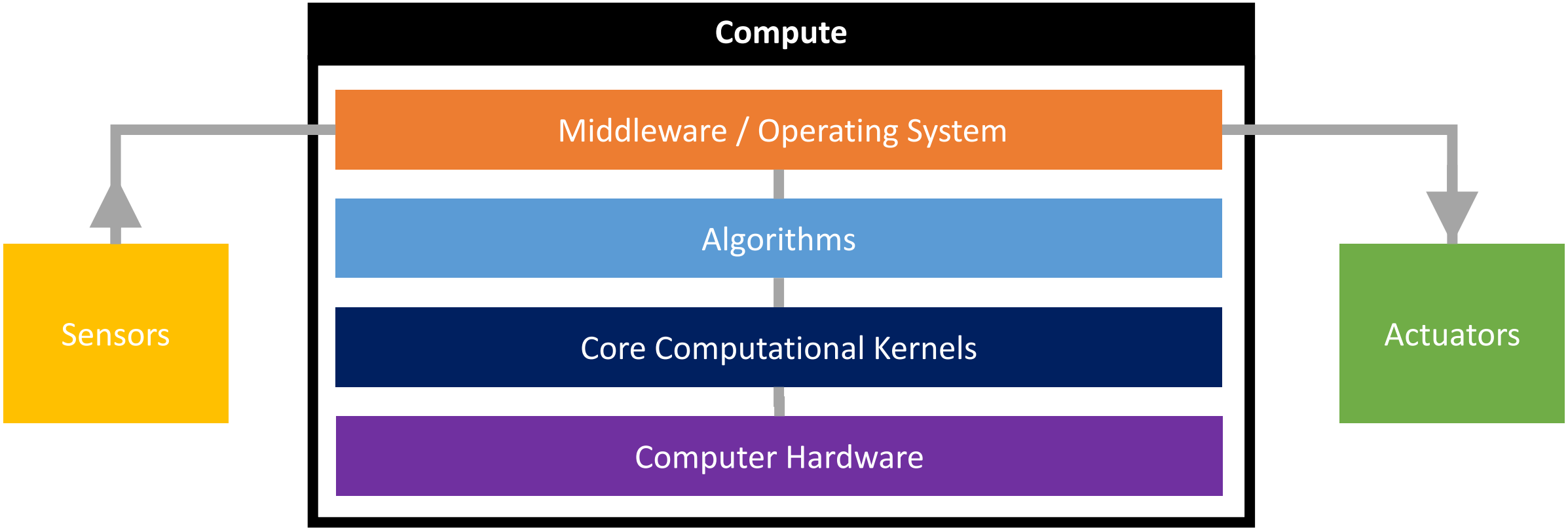


Robotics is a **BIG** space

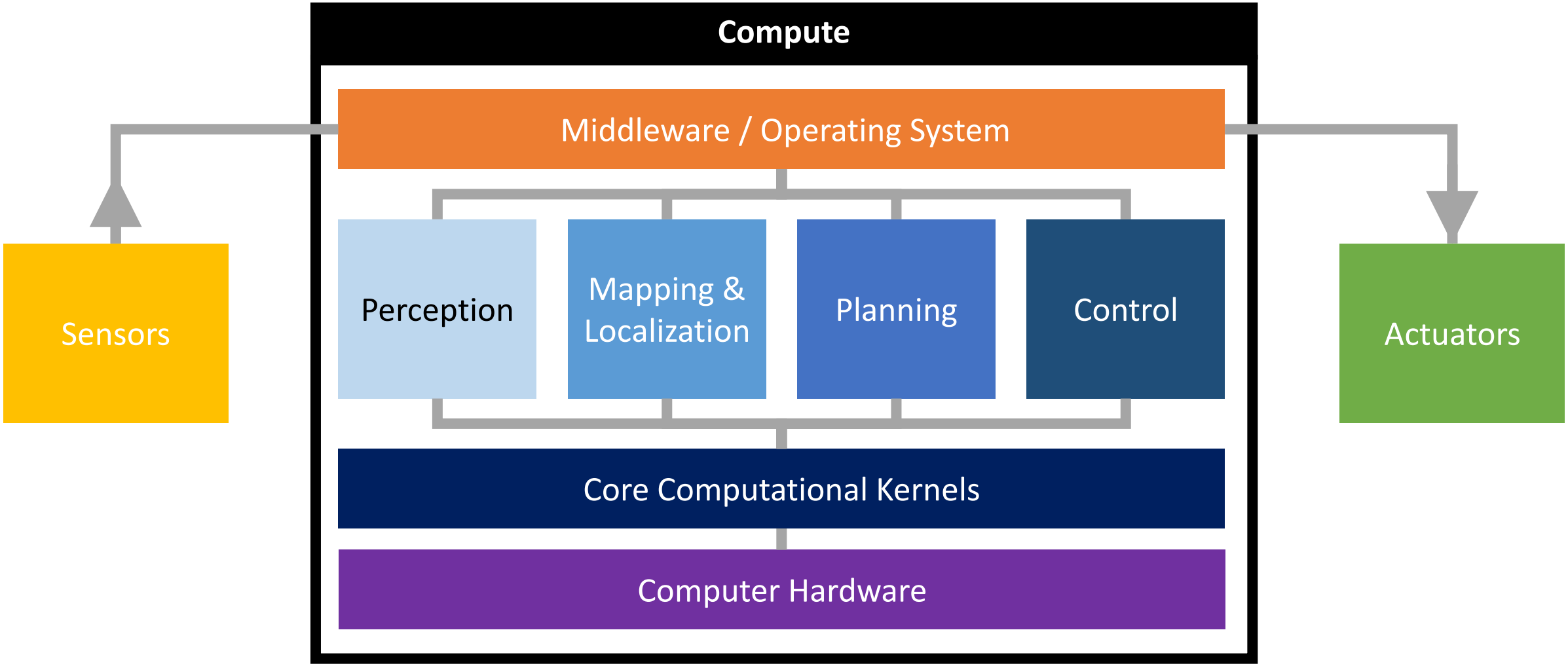
Sensors → Compute → Actuators



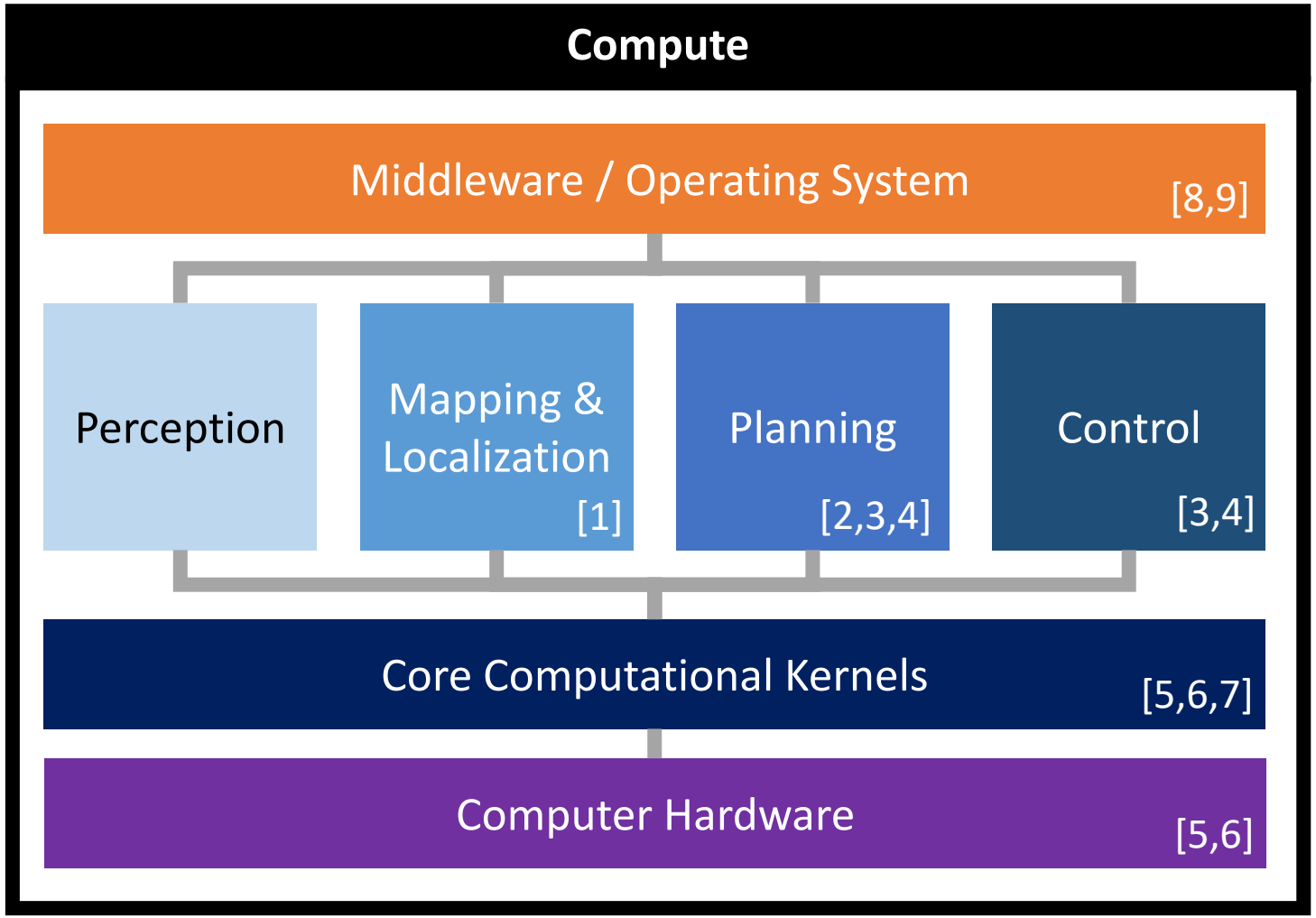
The Robotics Pipeline



The Robotics Pipeline



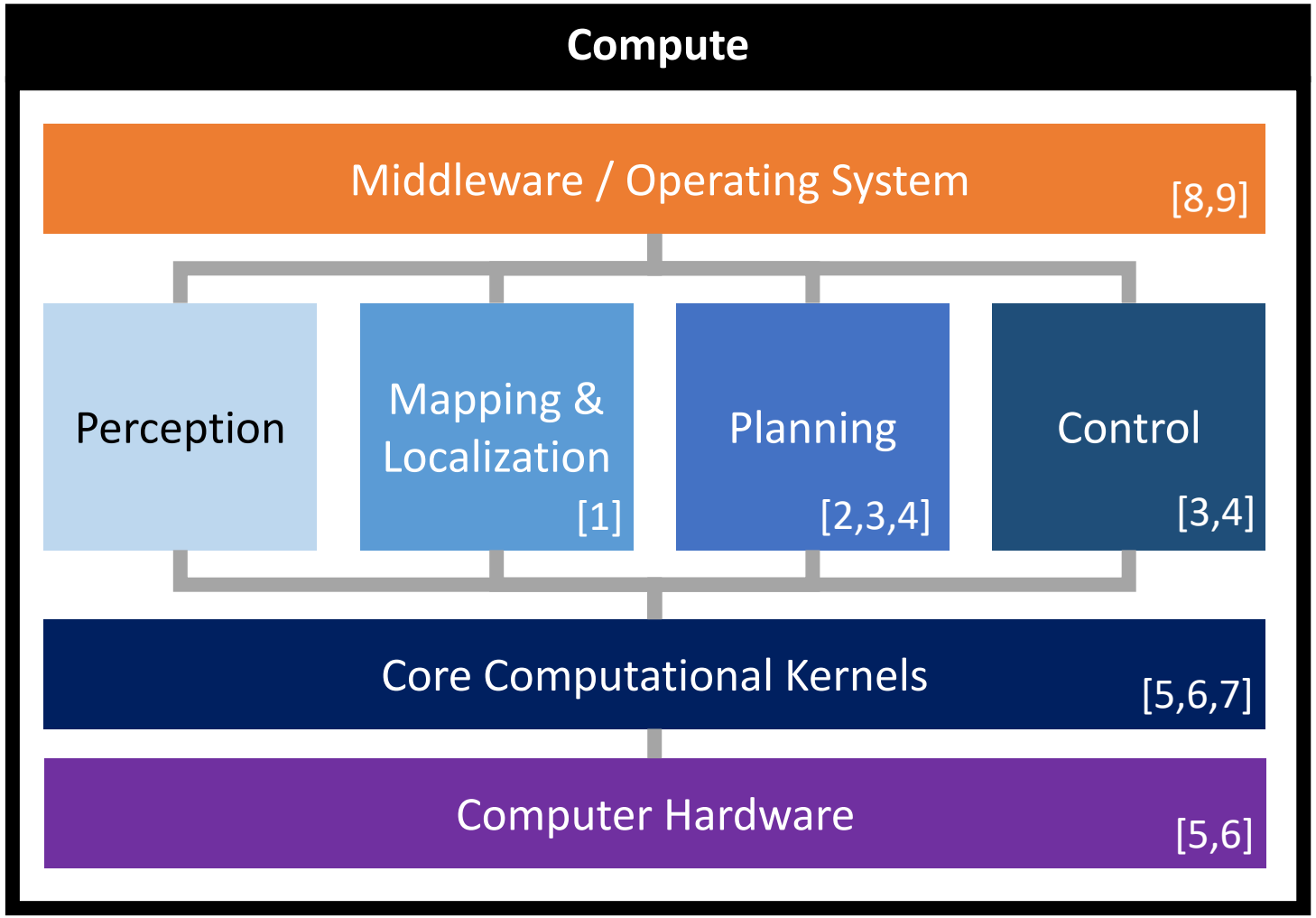
The Robotics Pipeline



[1] **B. Plancher**, C. D. Brumar, I. Brumar, L. Pentecost, S. Rama, D. Brooks. Application of Approximate Matrix Multiplication to Neural Networks and Distributed SLAM. HPEC 2019.
[2] **B. Plancher**, Z. Manchester, S. Kuindersma. Constrained Unscented Dynamic Programming. IROS 2017.
[3] **B. Plancher**, S. Kuindersma. A Performance Analysis of Parallel Differential Dynamic Programming on a GPU. WAFR 2018.
[4] **B. Plancher**, S. Kuindersma. Realtime Model Predictive Control using Parallel DDP on a GPU. In Online Optimal Control Workshop at ICRA 2019.
[5] **B. Plancher**, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.
[6] S. M. Neuman, **B. Plancher**, T. Bourgeat, T. Tambe, S. Devadas, V. Janapa Reddi. Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology. ASPLOS 2021.
[7] [Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.
[8] B. Boroujerdian, R. Ghosal, J. Cruz, **B. Plancher**, V. Janapa Reddi. RoboRun: A Robot Runtime to Exploit Spatial Heterogeneity. DAC 2021.
[9] [Under Review] B. Boroujerdian, H. Genc, S. Krishnan, P. Bardienus, B. Duisterhof, **B. Plancher**, K. Mansoorshahi, M. Almeida, A. Faust, V. Janapa Reddi. The Role of Compute in Autonomous Aerial Vehicles. TOCS, 2022.
[10] **B. Plancher**, V. Janapa Reddi. TinyMLedu: The Tiny Machine Learning Open Education Initiative. SIGCSE 2022.
[11] V. Janapa Reddi, **B. Plancher**, S. Kennedy, L. Moroney, P. Warden, L. Suzaki, A. Agarwal, C. Banbury, M. Banzi, M. Bennett, B. Brown, S. Chitlangia, R. Ghosal, S. Grafman, R. Jaeger, S. Krishnan, M. Lam, D. Leiker, C. Mann, M. Mazumder, D. Pajak , D. Ramaprasad , J. E. Smith, M. Stewart, D. Tingley. Widening Access to Applied Machine Learning with TinyML. HDSR 2022.
[12] S. Karaman, A. Anders, M. Boulet, J. Connor, K. Gregson, W. Guerra, O. Guldner, M. Mohamoud, **B. Plancher**, R. Shin, J. Vivilecchia. Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT. ISEC 2017.

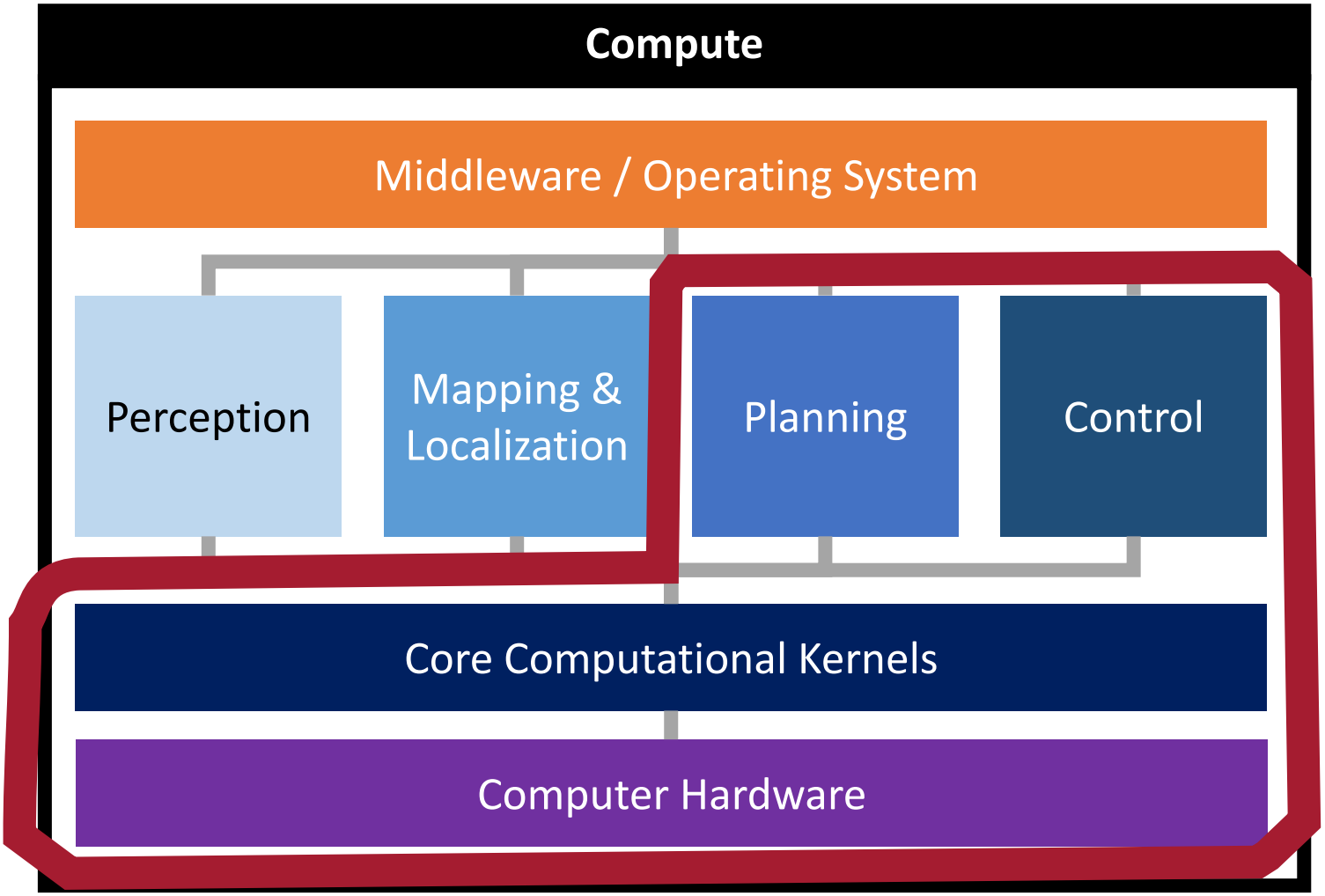
The Robotics Pipeline

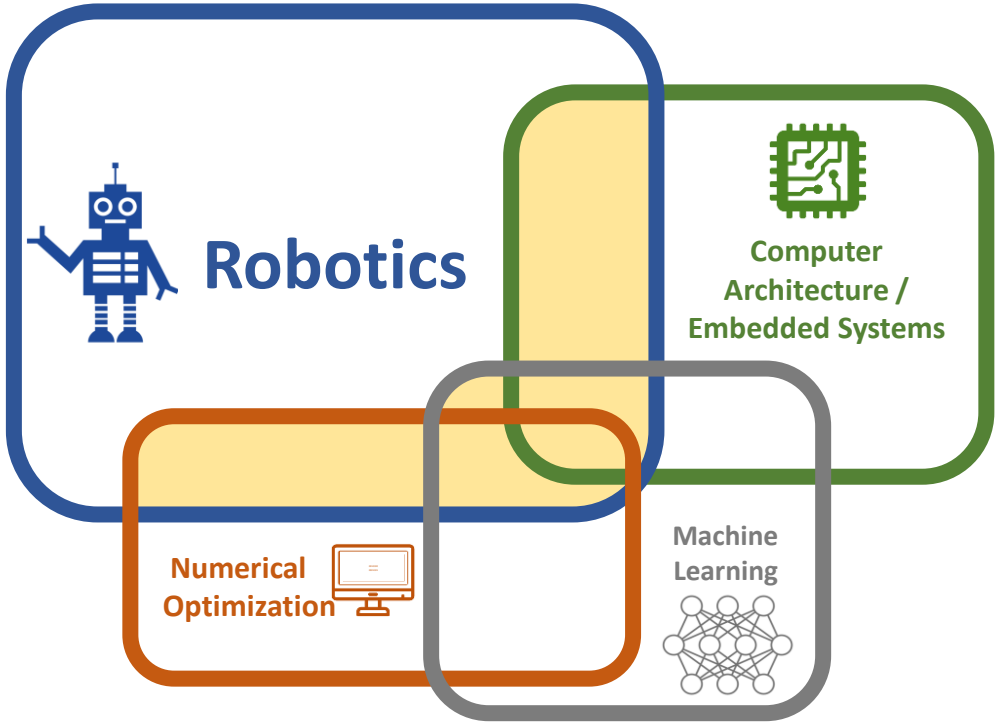
CS Education
[10,11,12]



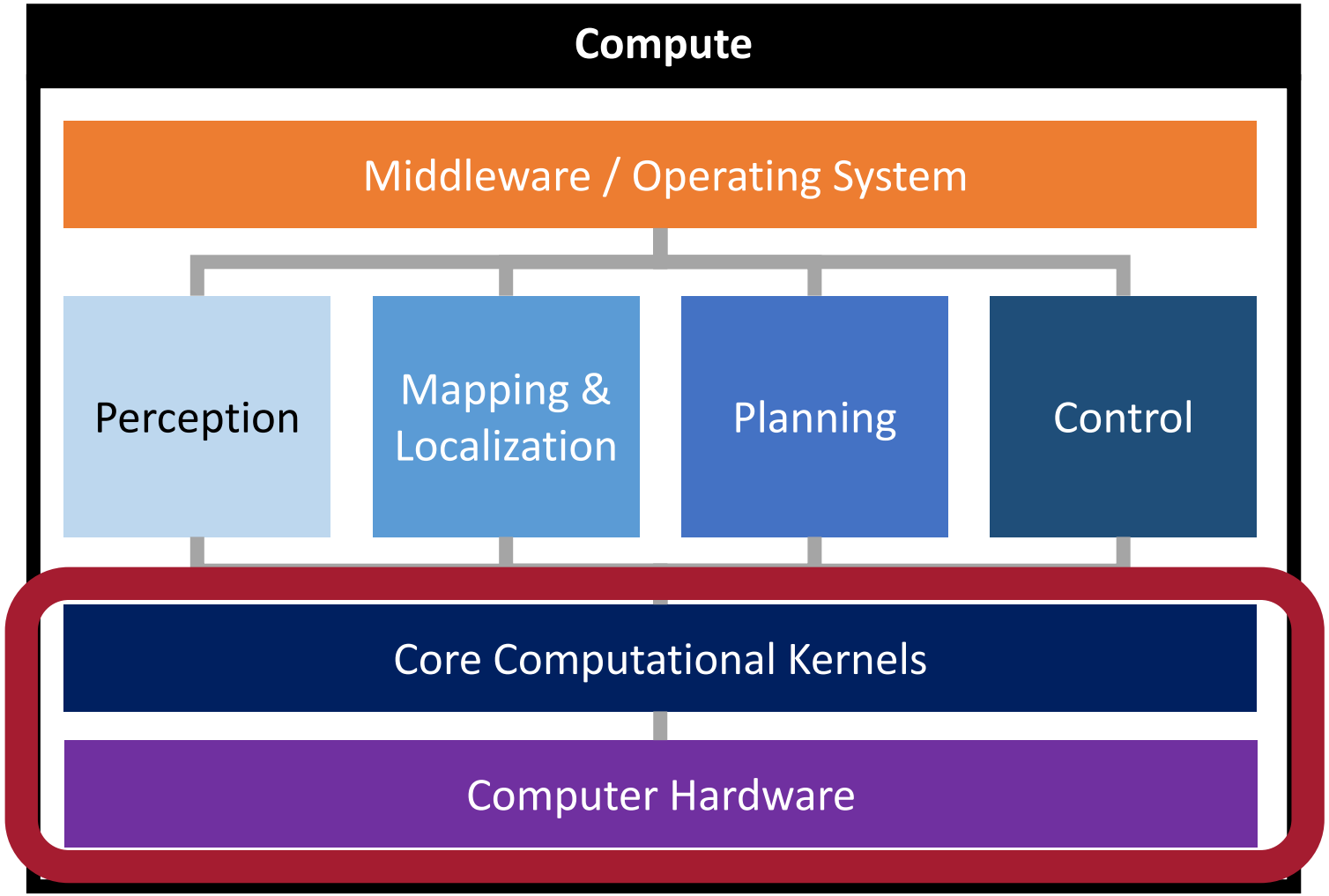
[1] **B. Plancher**, C. D. Brumar, I. Brumar, L. Pentecost, S. Rama, D. Brooks. Application of Approximate Matrix Multiplication to Neural Networks and Distributed SLAM. HPEC 2019.
[2] **B. Plancher**, Z. Manchester, S. Kuindersma. Constrained Unscented Dynamic Programming. IROS 2017.
[3] **B. Plancher**, S. Kuindersma. A Performance Analysis of Parallel Differential Dynamic Programming on a GPU. WAFR 2018.
[4] **B. Plancher**, S. Kuindersma. Realtime Model Predictive Control using Parallel DDP on a GPU. In Online Optimal Control Workshop at ICRA 2019.
[5] **B. Plancher**, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.
[6] S. M. Neuman, **B. Plancher**, T. Bourgeat, T. Tambe, S. Devadas, V. Janapa Reddi. Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology. ASPLOS 2021.
[7] [Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.
[8] B. Boroujerdian, R. Ghosal, J. Cruz, **B. Plancher**, V. Janapa Reddi. RoboRun: A Robot Runtime to Exploit Spatial Heterogeneity. DAC 2021.
[9] [Under Review] B. Boroujerdian, H. Genc, S. Krishnan, P. Bardienus, B. Duisterhof, **B. Plancher**, K. Mansoorshahi, M. Almeida, A. Faust, V. Janapa Reddi. The Role of Compute in Autonomous Aerial Vehicles. TOCS, 2022.
[10] **B. Plancher**, V. Janapa Reddi. TinyMLedu: The Tiny Machine Learning Open Education Initiative. SIGCSE 2022.
[11] V. Janapa Reddi, **B. Plancher**, S. Kennedy, L. Moroney, P. Warden, L. Suzaki, A. Agarwal, C. Banbury, M. Banzi, M. Bennett, B. Brown, S. Chitlangia, R. Ghosal, S. Grafman, R. Jaeger, S. Krishnan, M. Lam, D. Leiker, C. Mann, M. Mazumder, D. Pajak , D. Ramaprasad , J. E. Smith, M. Stewart, D. Tingley. Widening Access to Applied Machine Learning with TinyML. HDSR 2022.
[12] S. Karaman, A. Anders, M. Boulet, J. Connor, K. Gregson, W. Guerra, O. Guldner, M. Mohamoud, **B. Plancher**, R. Shin, J. Vivilecchia. Project-based, collaborative, algorithmic robotics for high school students: Programming self-driving race cars at MIT. ISEC 2017.

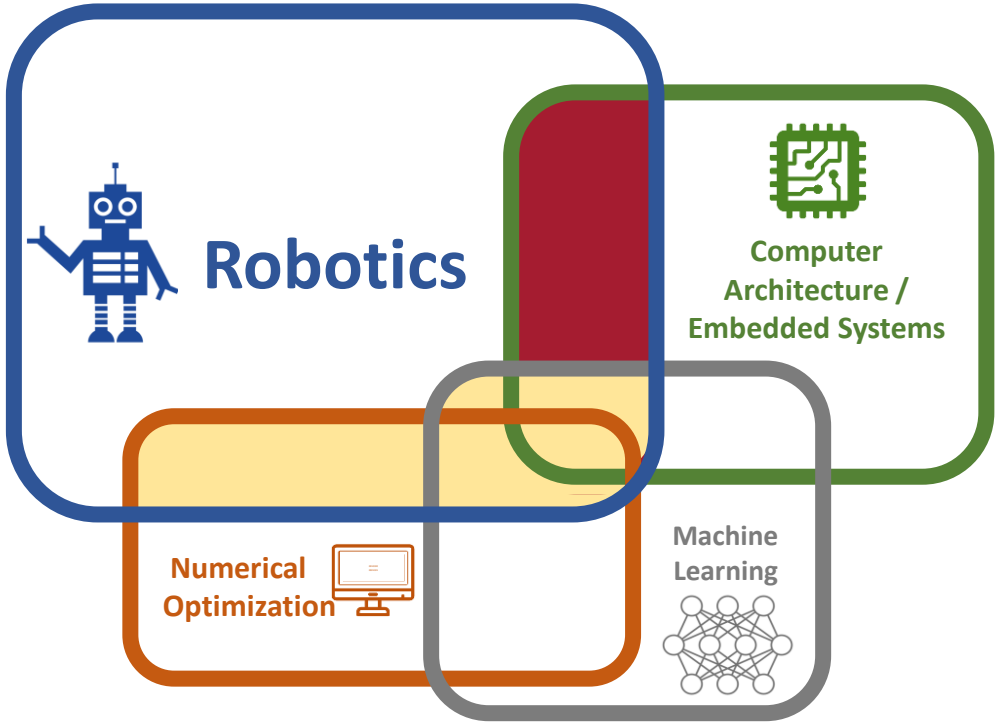
I work at the intersection of robotics and adjacent fields





I work at the intersection of robotics and adjacent fields





Hardware Acceleration for Realtime Robotics

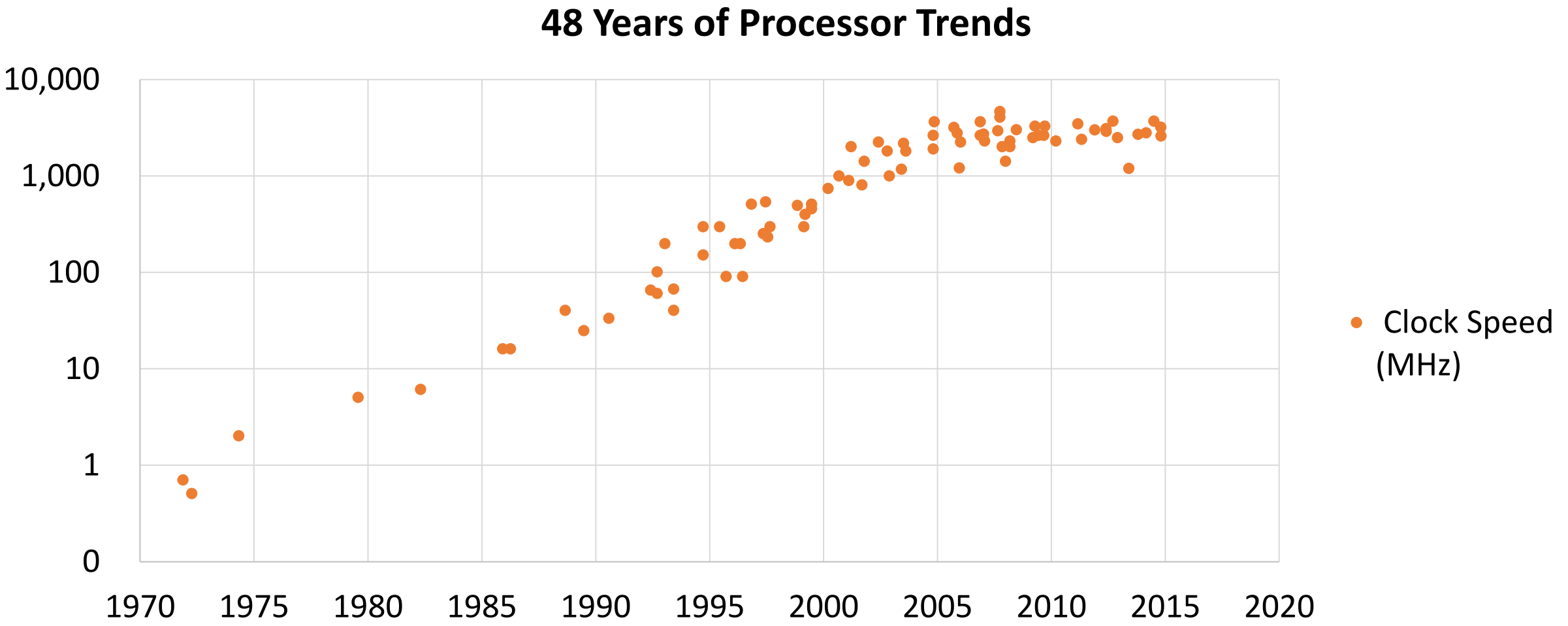
A Quick Overview of (And My Approach To) Robotics

A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

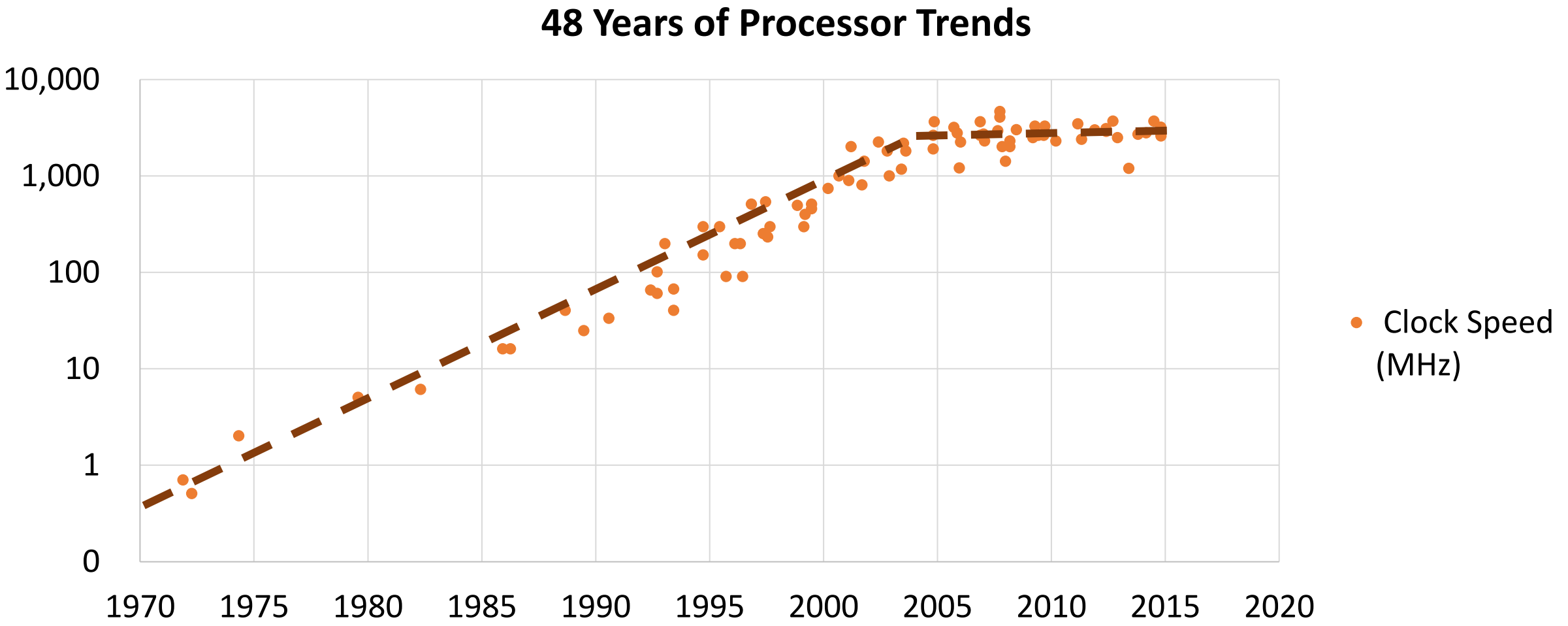
The Future of Accelerating Robotics

CPUs are not getting faster…



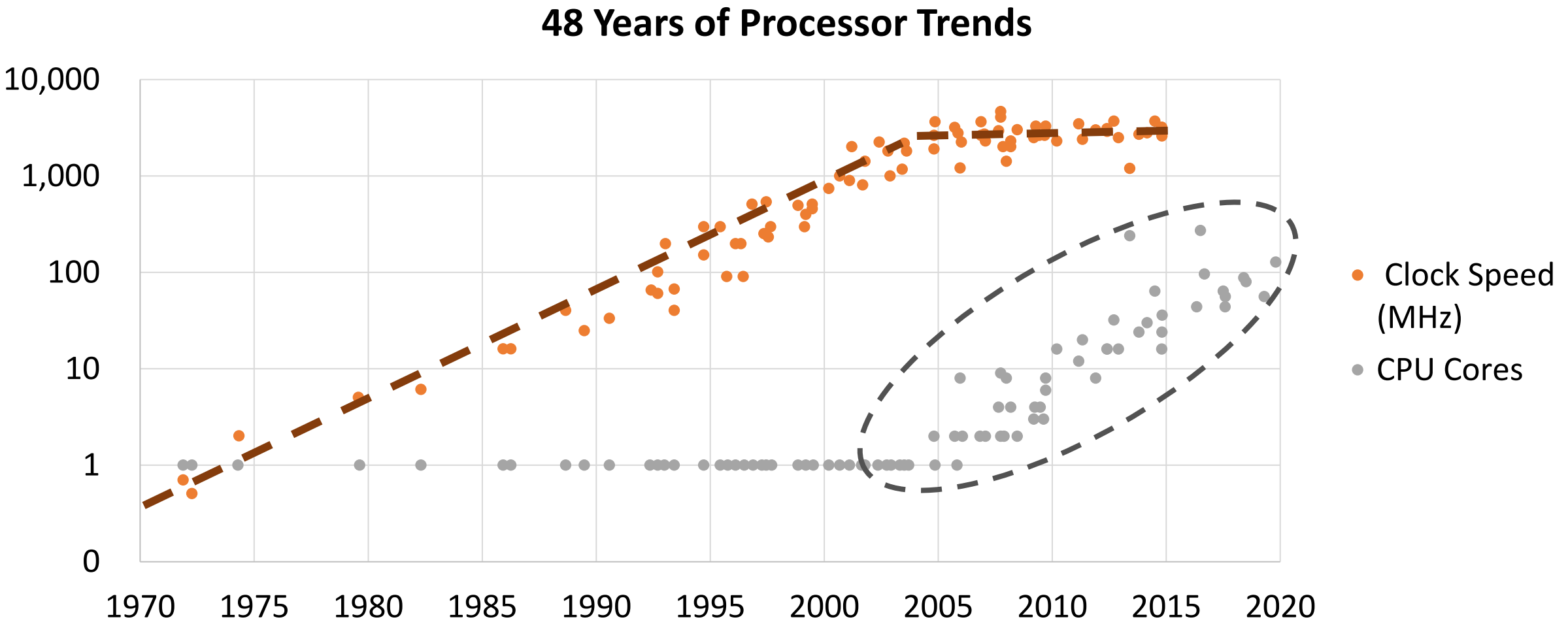
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp https://github.com/karlrupp/microprocessor-trend-data

CPUs are not getting faster…



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp https://github.com/karlrupp/microprocessor-trend-data

… instead, we need to focus on Parallelism

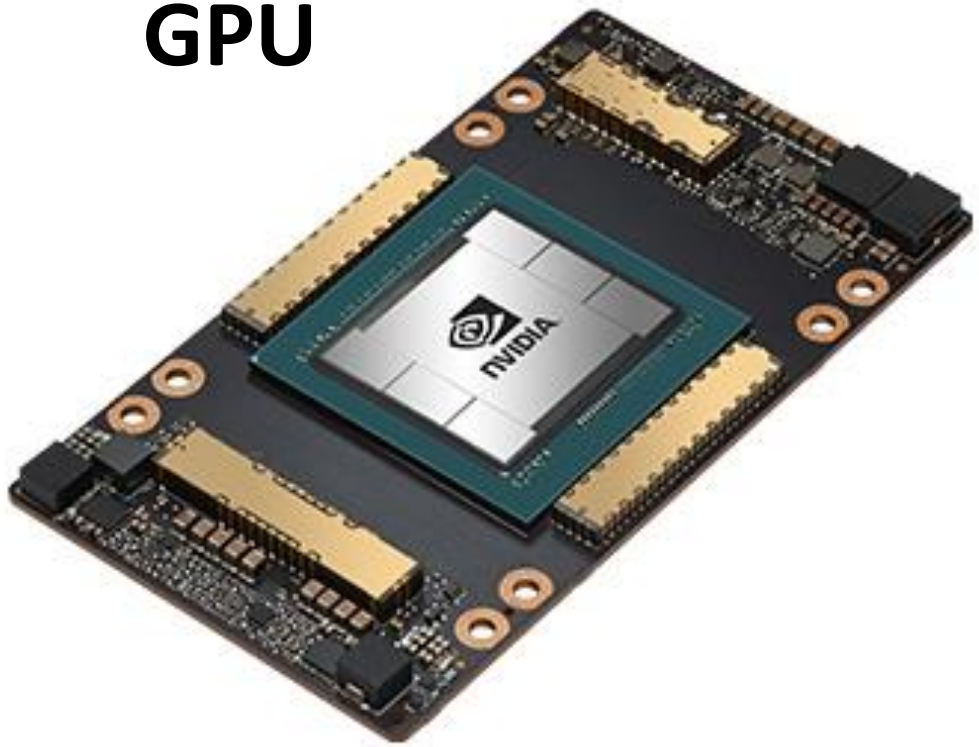


Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp https://github.com/karlrupp/microprocessor-trend-data

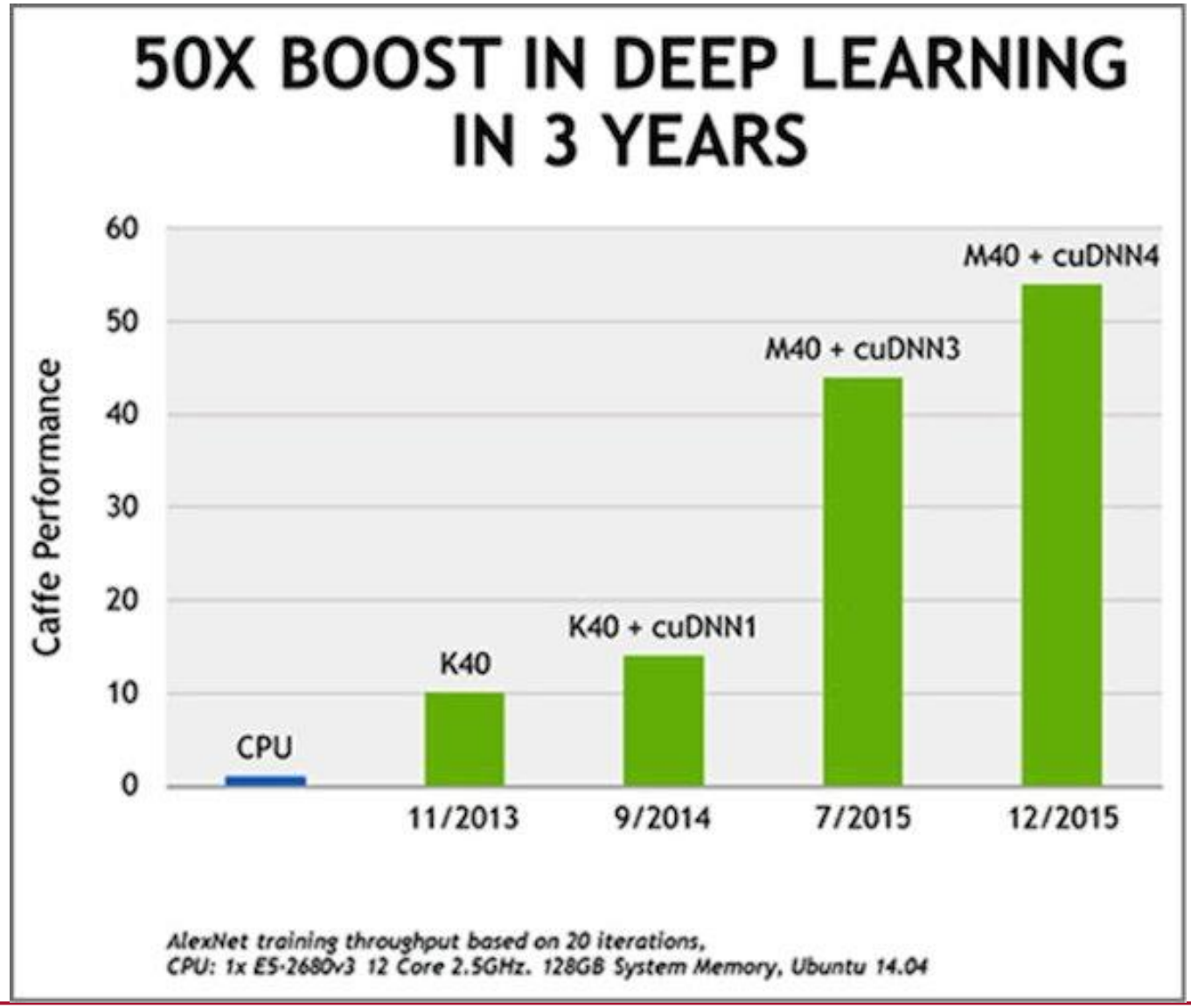
… instead, we need to focus on Parallelism

CPU

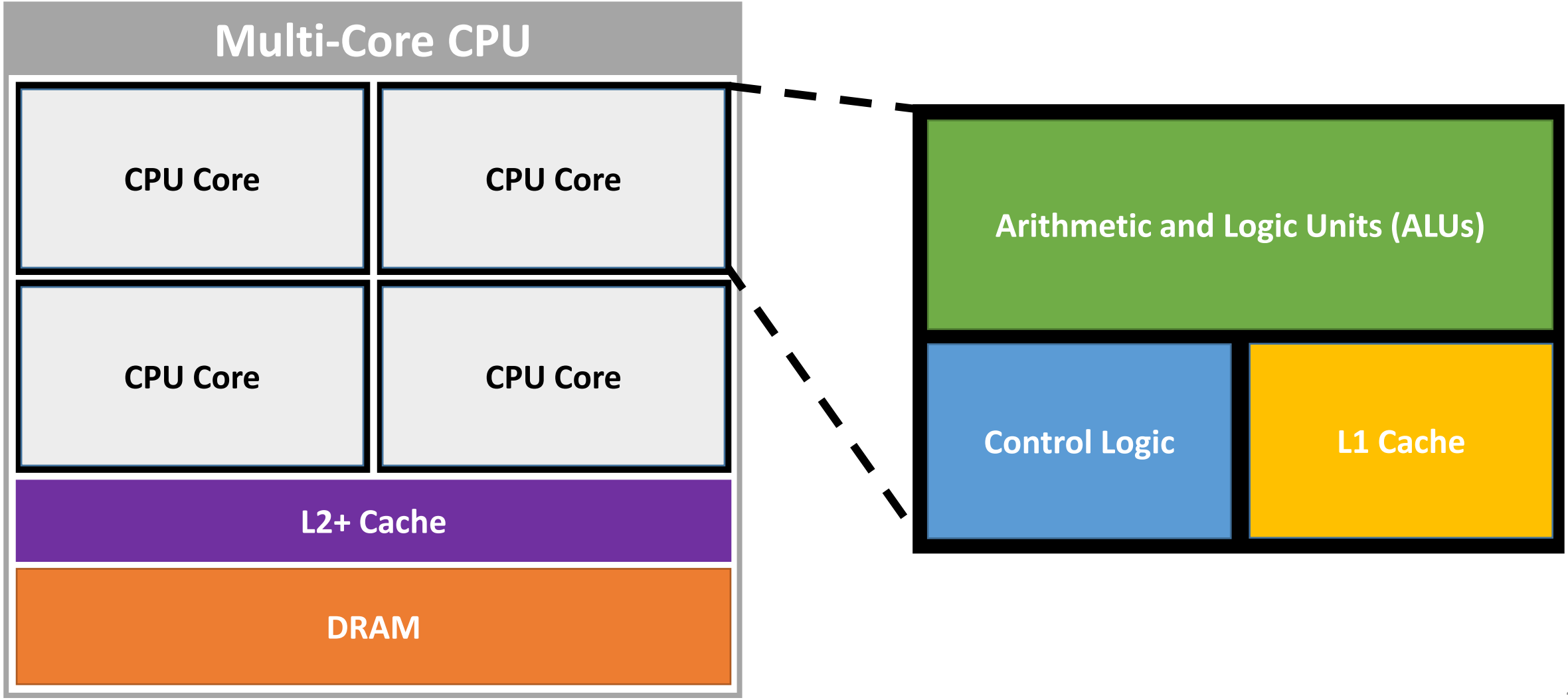
GPU



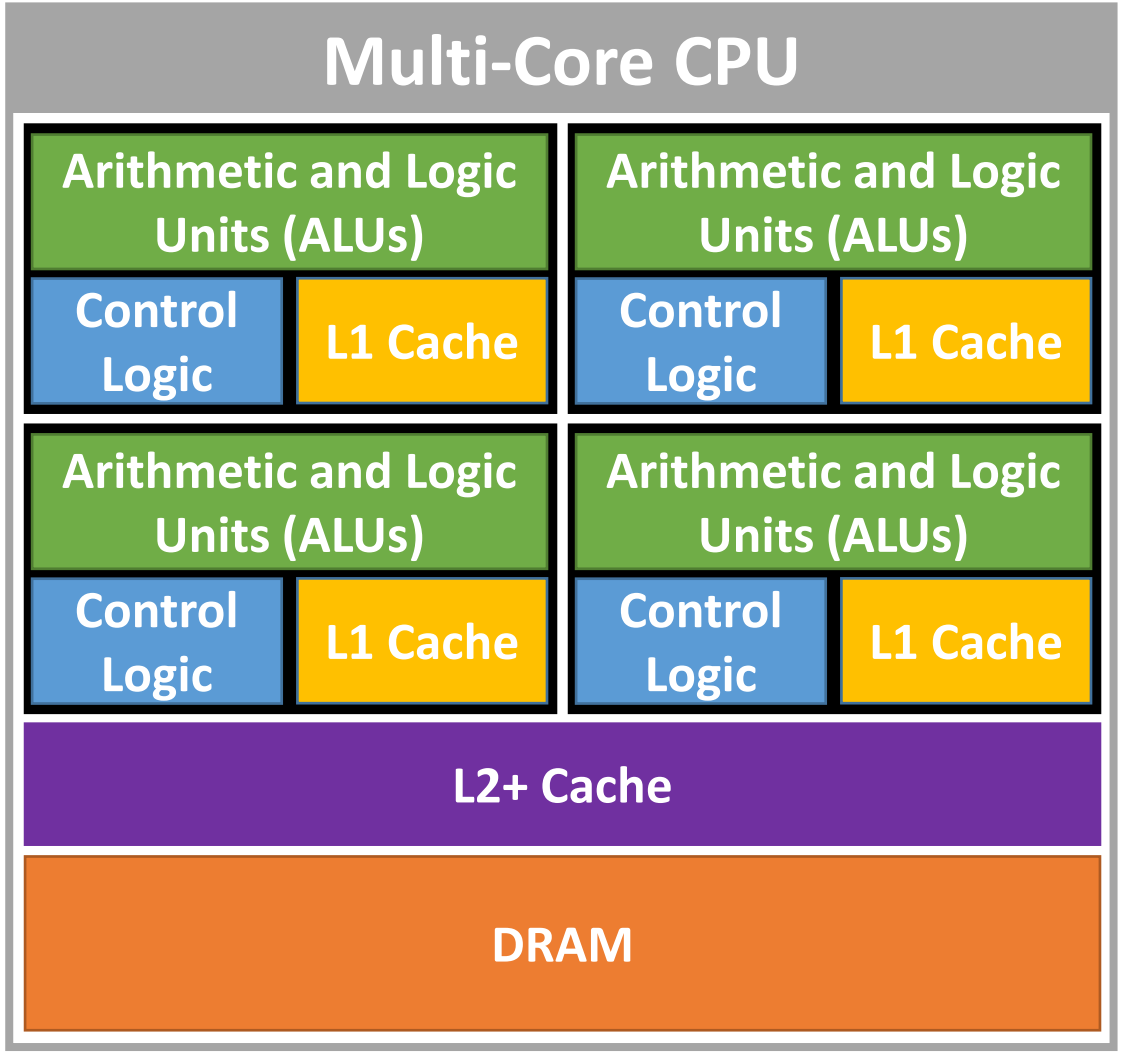
CPUs and GPUs have fundamentally different strengths and weaknesses



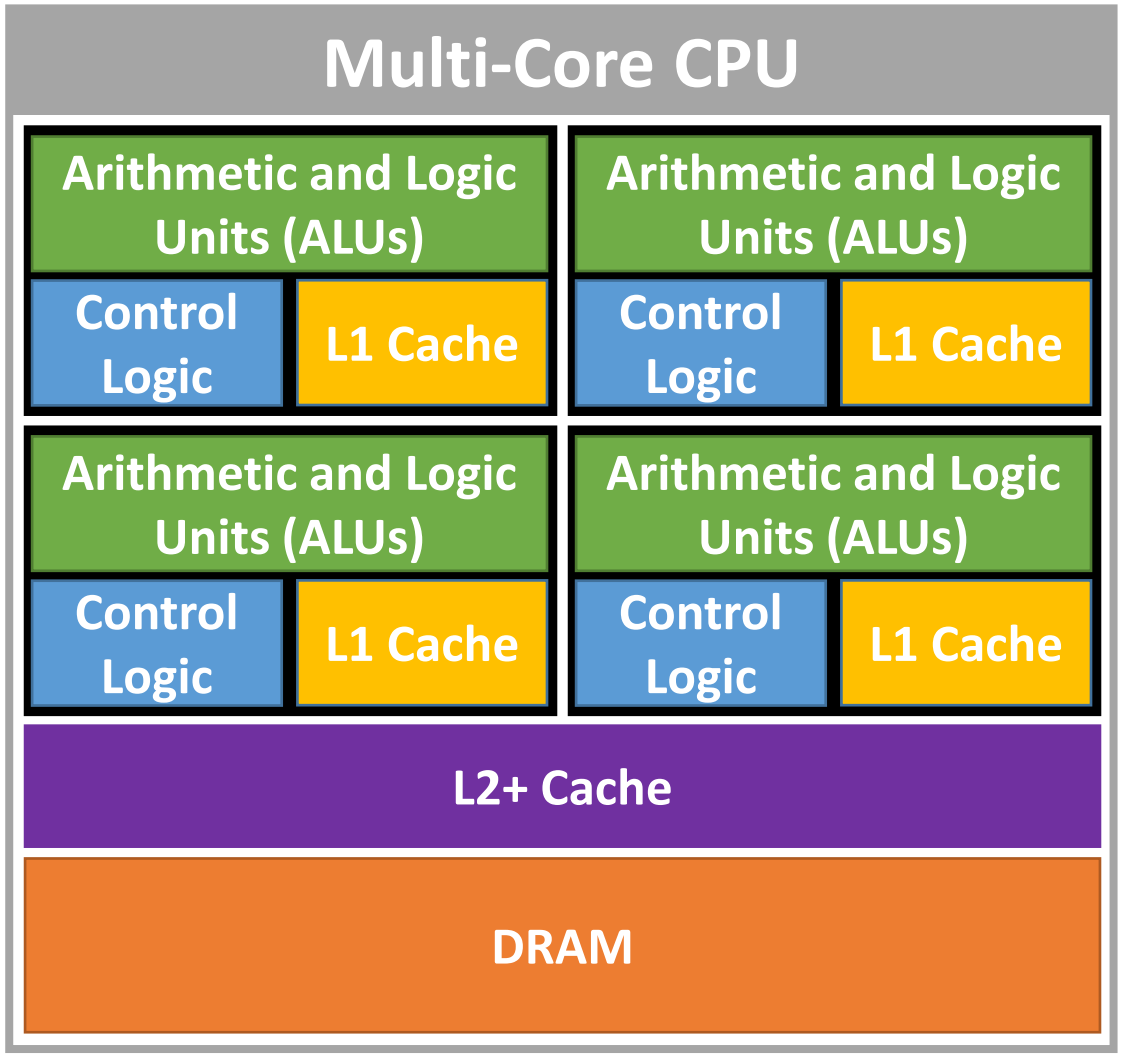
CPUs and GPUs have fundamentally different strengths and weaknesses

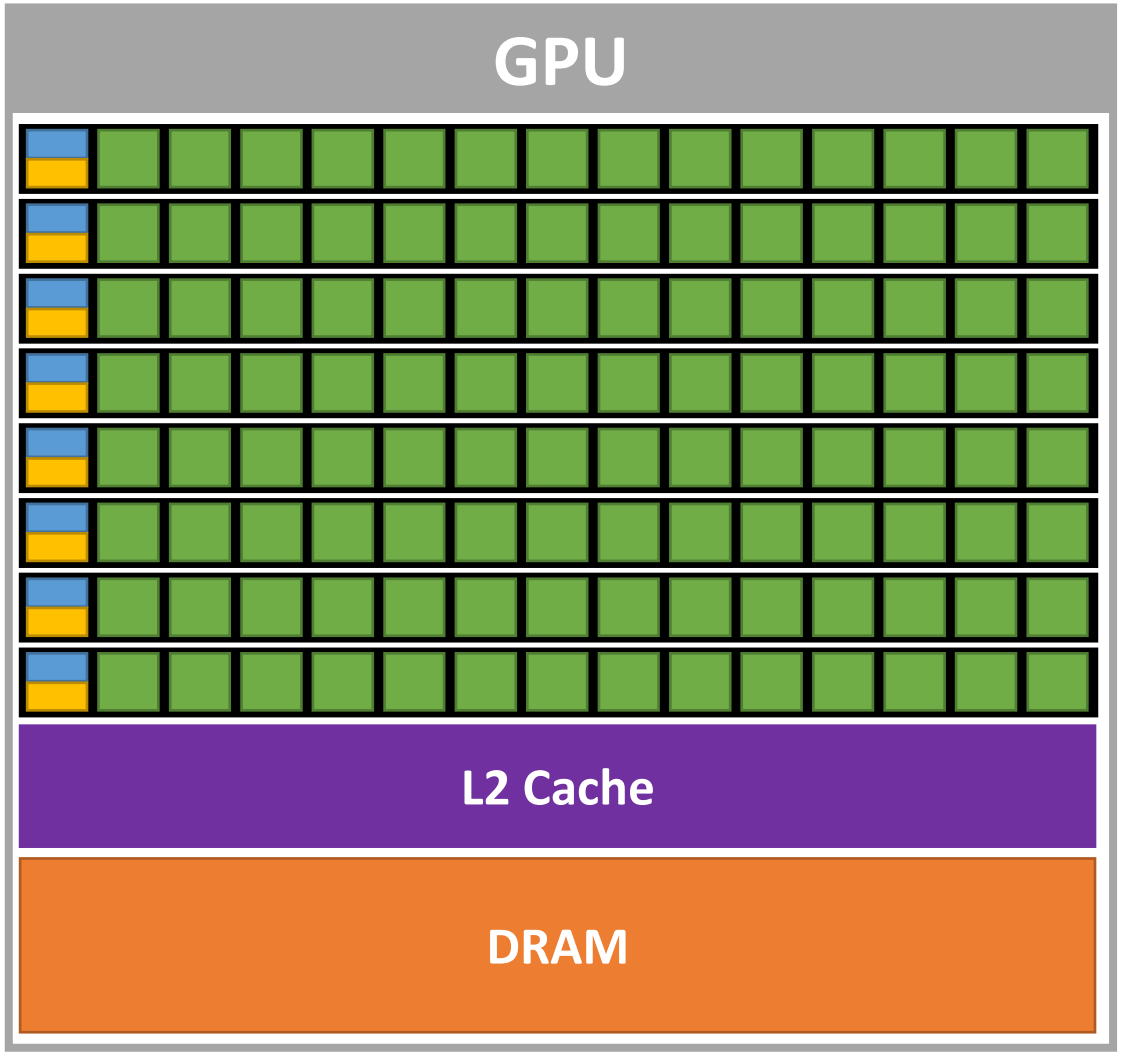


CPUs and GPUs have fundamentally different strengths and weaknesses



CPUs and GPUs have fundamentally different strengths and weaknesses





Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

The Future of Accelerating Robotics

Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

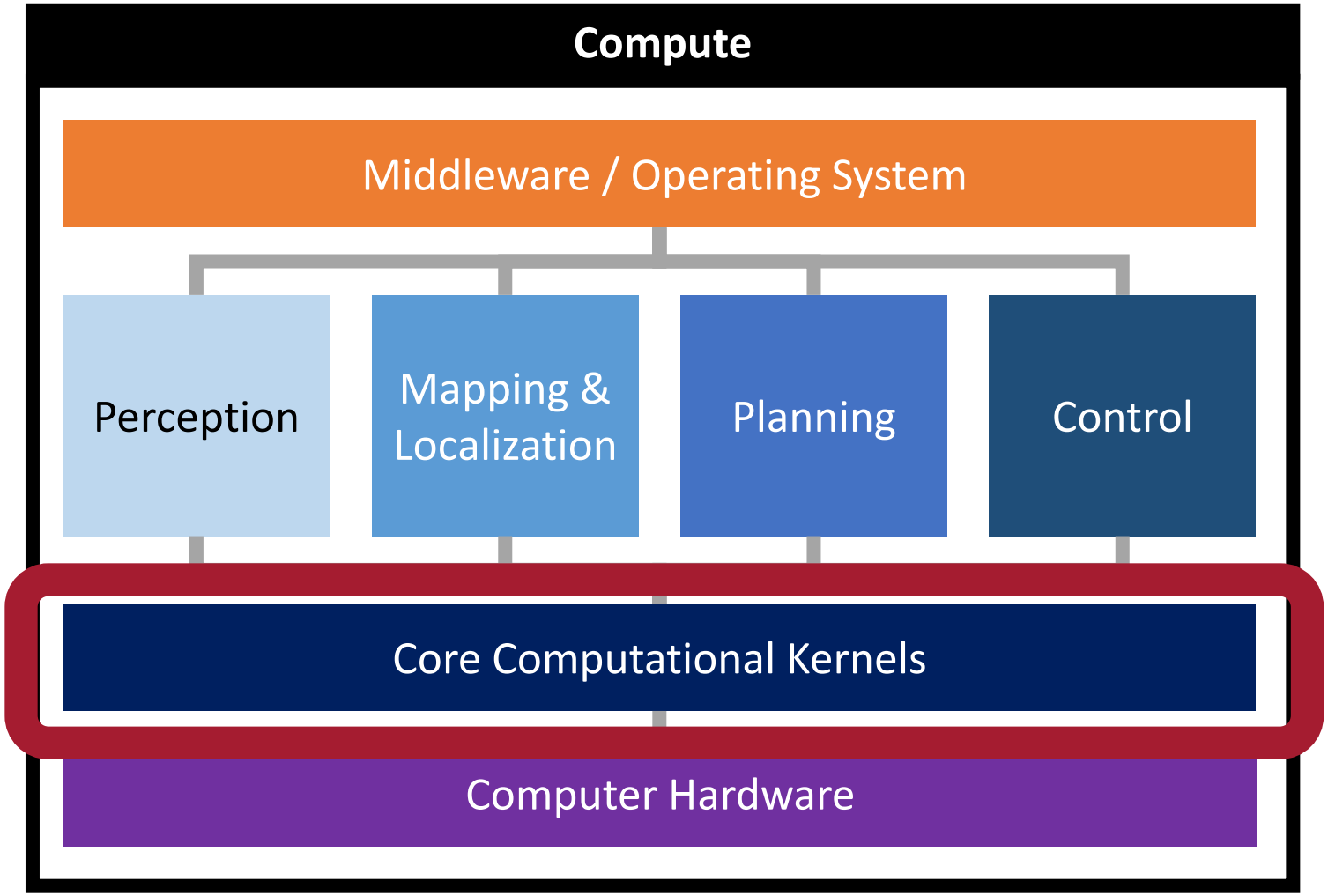
A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

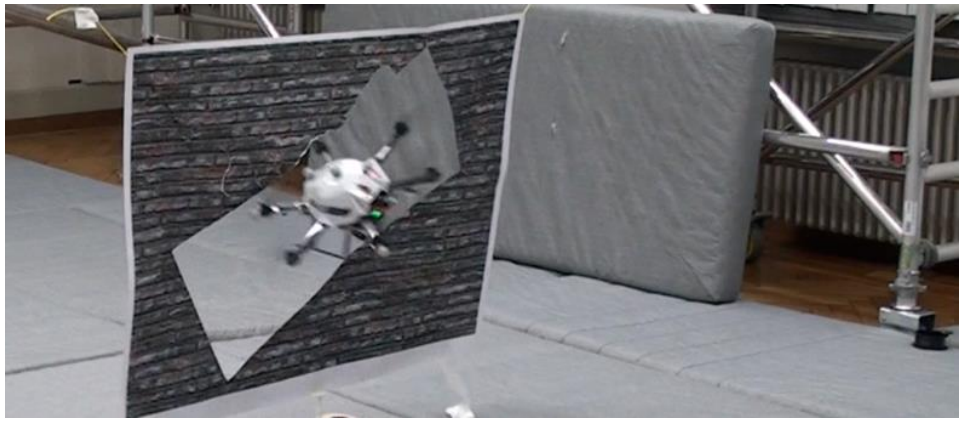
- Why Accelerate Rigid Body Dynamics?
- Accelerating Rigid Body Dynamics on the GPU
- Moving Beyond the GPU

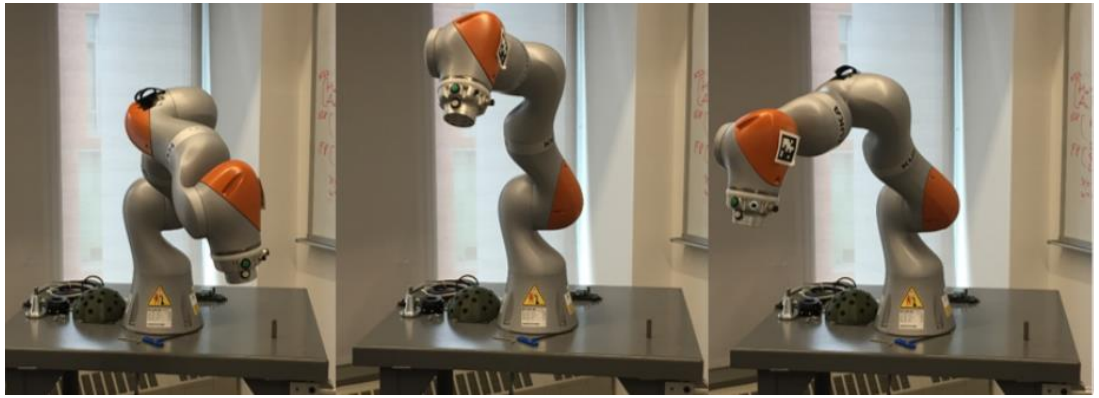
The Future of Accelerating Robotics

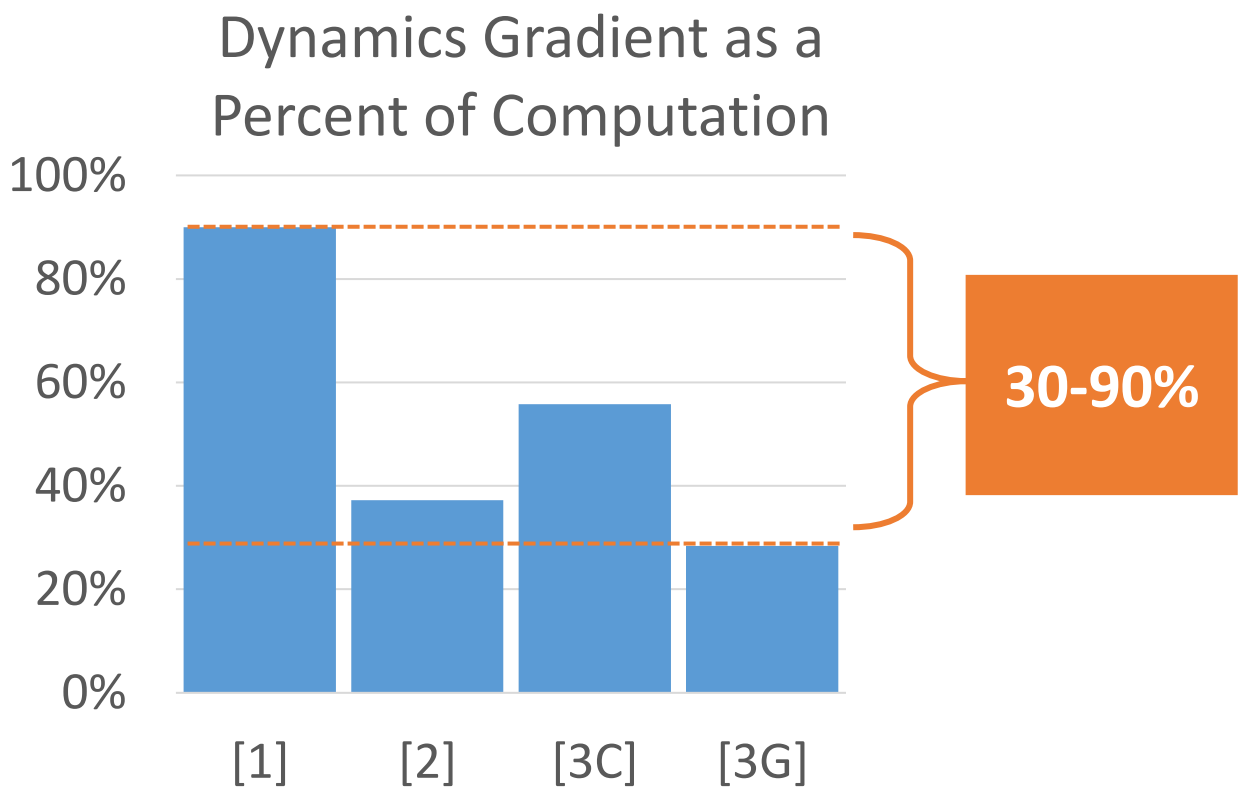
Why Rigid Body Dynamics



Rigid Body Dynamics Gradients are a bottleneck for planning and control







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

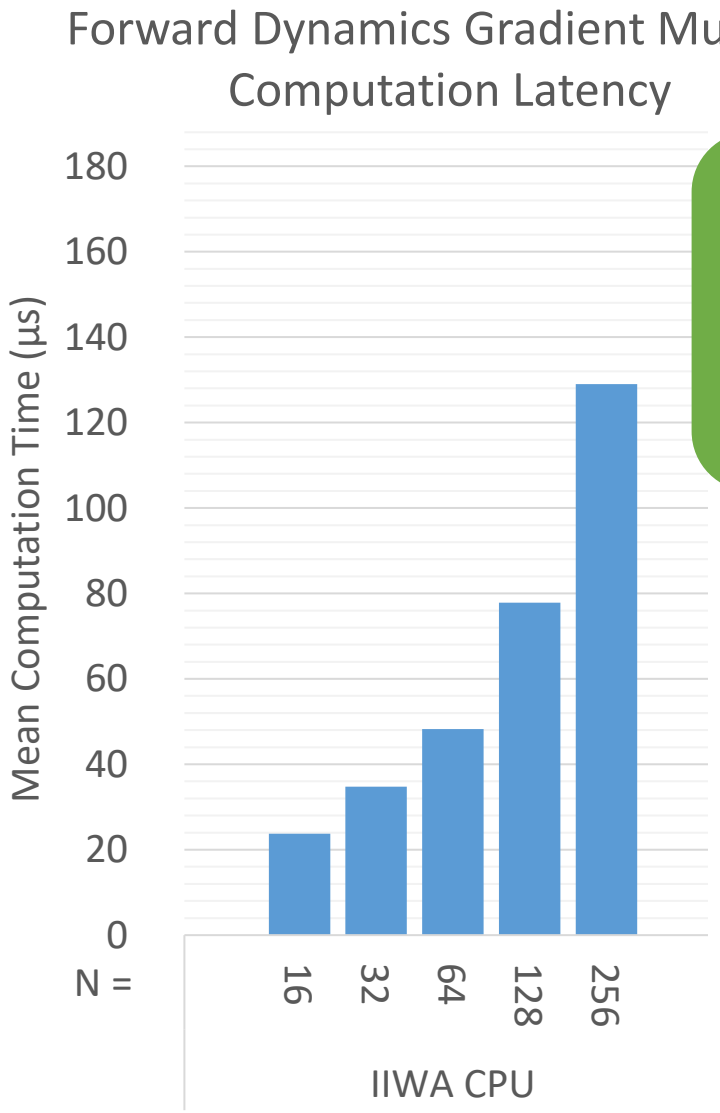
[2] M. Neunert, et al., "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking," ICRA 2016

[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

Why GPU Acceleration for Rigid Body Dynamics?

Planning and Control Algorithms often require many simultaneous computations of Rigid Body Dynamics

Why GPU Acceleration for Rigid Body Dynamics?



This can become very time consuming quickly! And is naturally parallel!

[Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.

Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

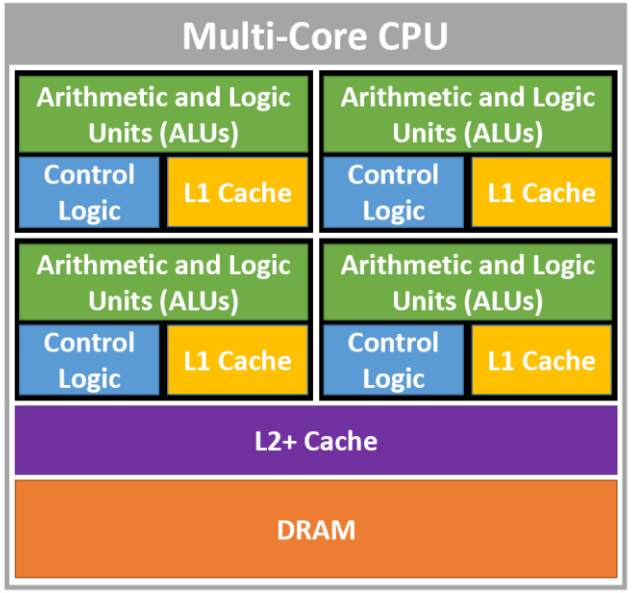
A Crash Course in (GPU) Architecture

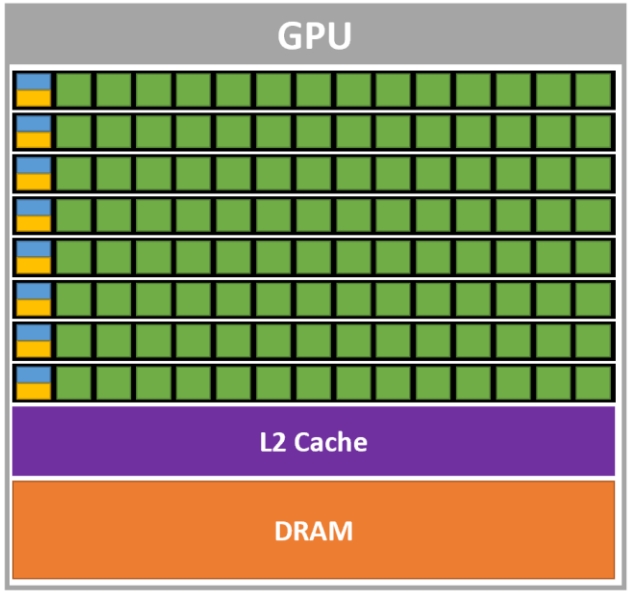
A Case Study in Accelerating Rigid Body Dynamics

- Why Accelerate Rigid Body Dynamics?
- Accelerating Rigid Body Dynamics on the GPU
- Moving Beyond the GPU

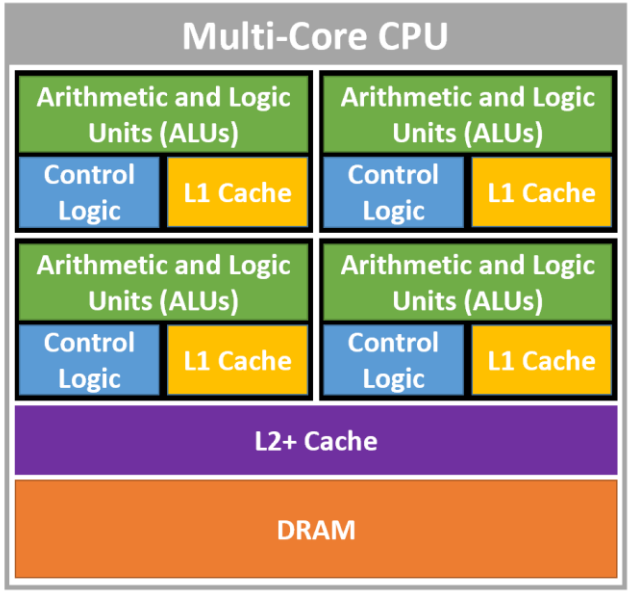
The Future of Accelerating Robotics

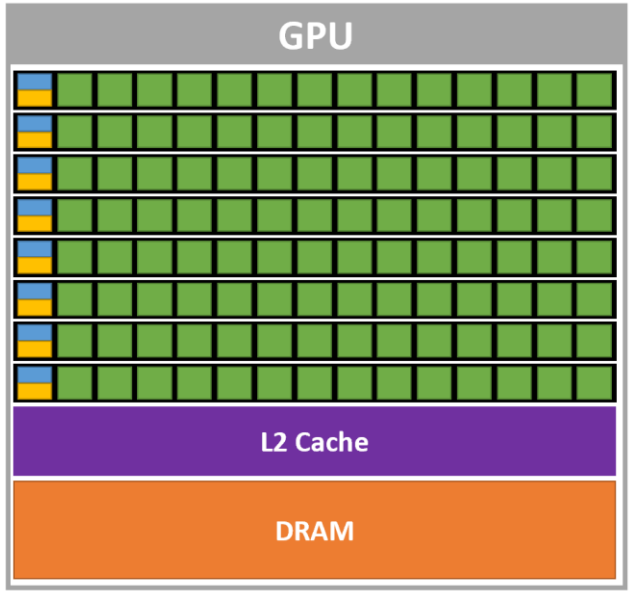
How do we Accelerate Rigid Body Dynamics on a GPU?





How do we Accelerate Rigid Body Dynamics on a GPU?



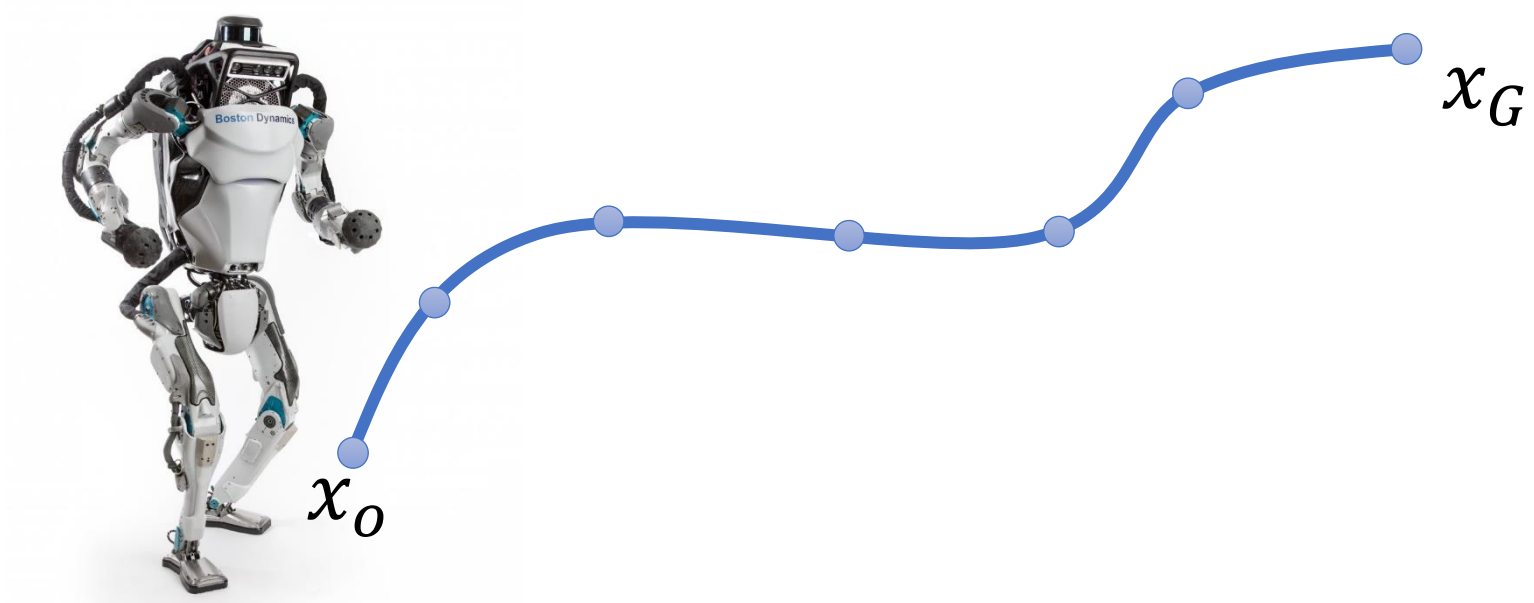


Hardware-Software Co-Design

High performance code needs to be **refactored** to take advantage of **different hardware** computational strengths and weaknesses

Algorithmic Features of the Gradient of Rigid Body Dynamics as a step of a planning and control algorithm

Algorithmic Feature	CPU	GPU
Parallelism	Bad	Good



Algorithmic Features of the Gradient of Rigid Body Dynamics as a step of a planning and control algorithm

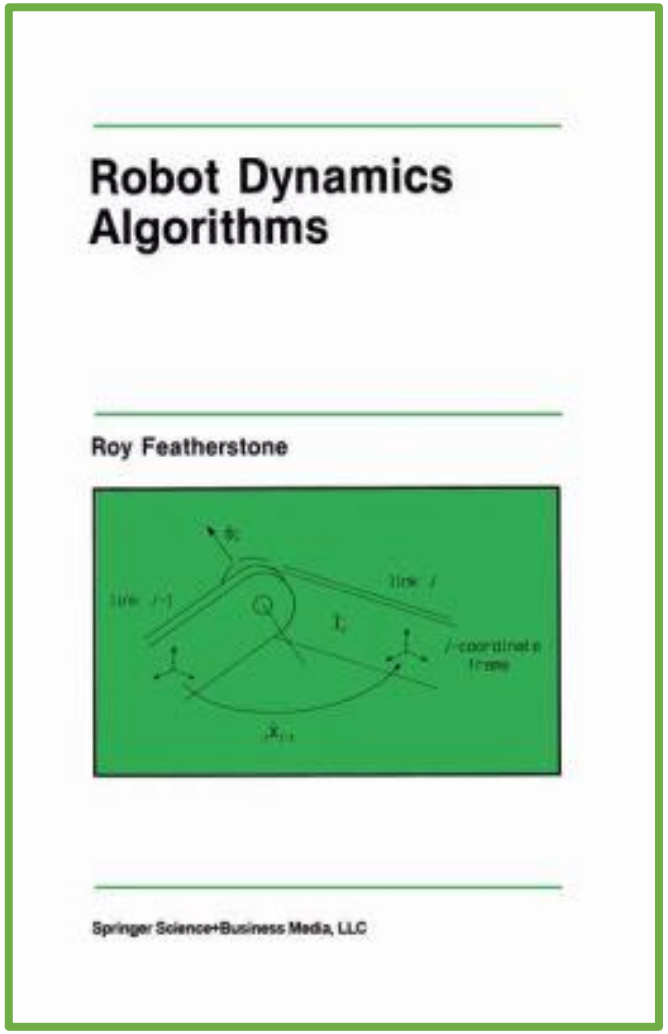
Algorithmic Feature	CPU	GPU
Parallelism	Bad	Good
?		

What is Rigid Body Dynamics?

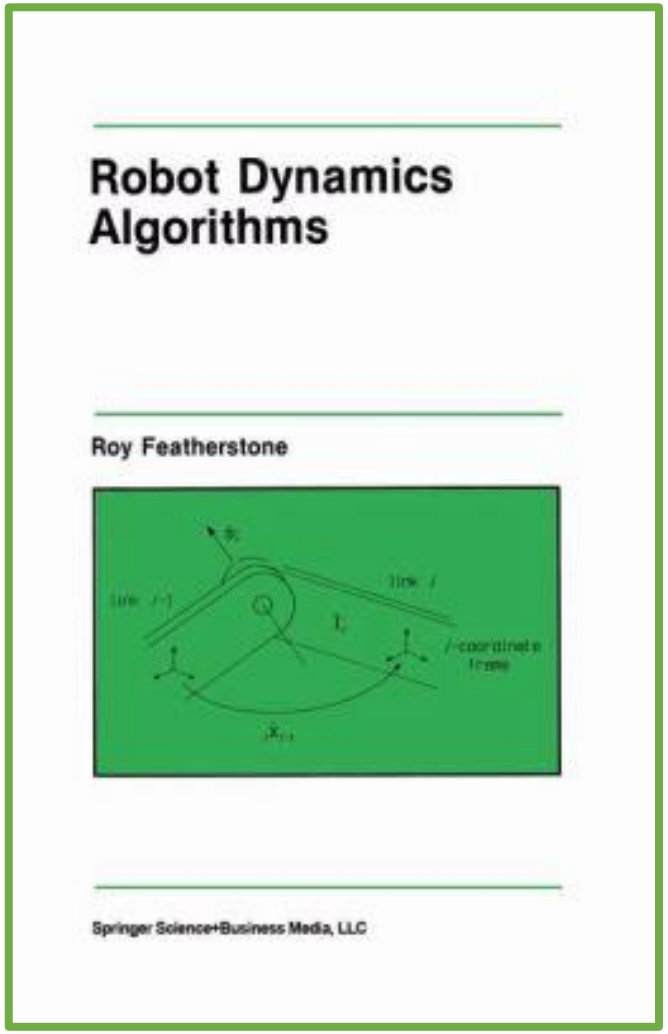
What is Rigid Body Dynamics?

Physics

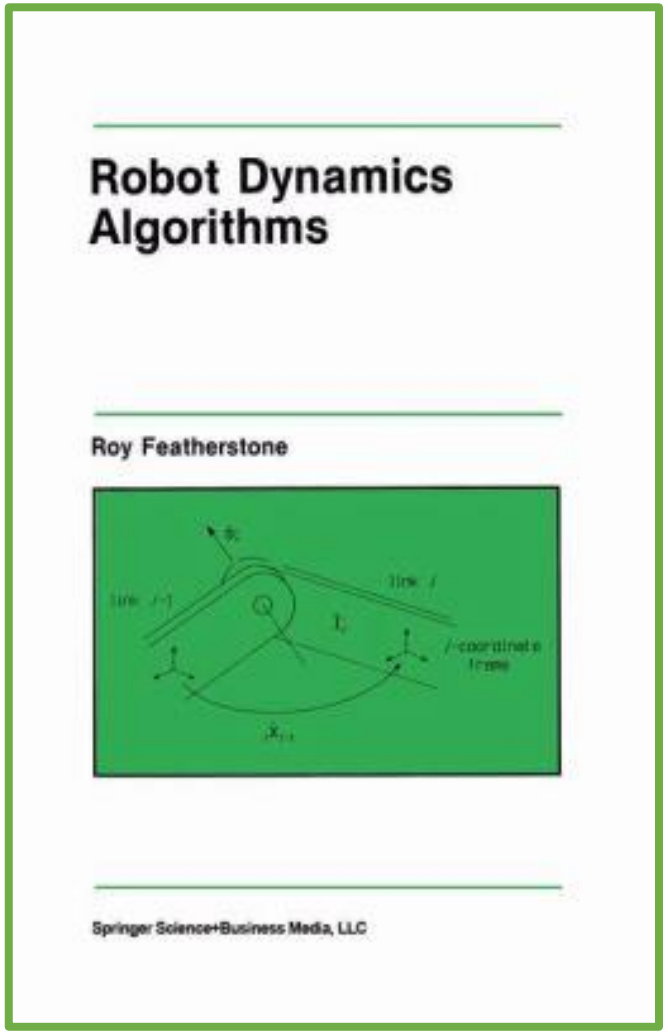
What is Rigid Body Dynamics?



What is Rigid Body Dynamics?

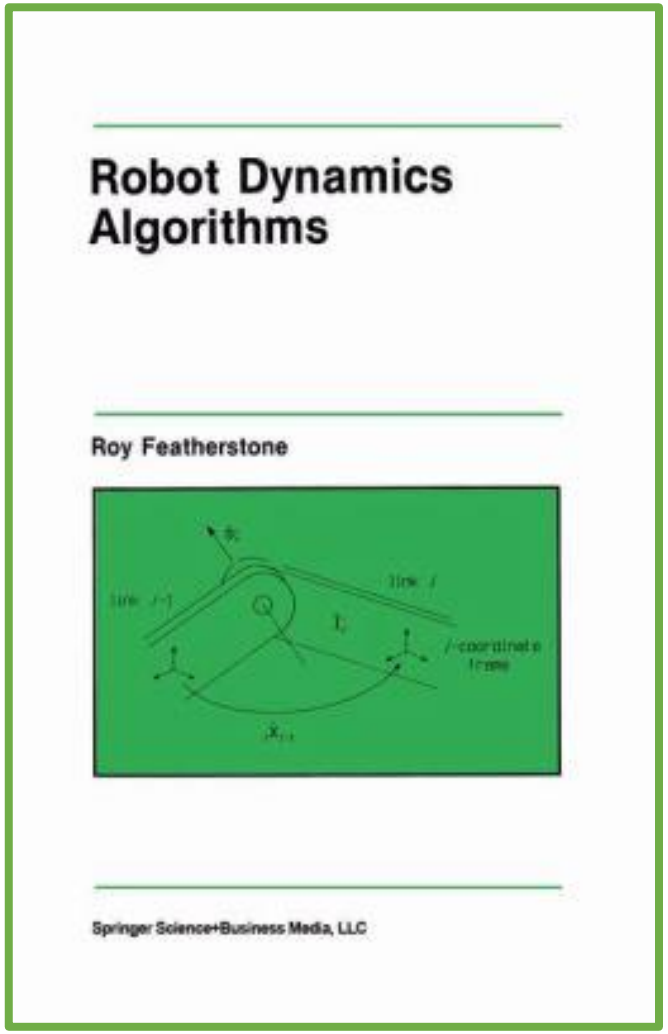


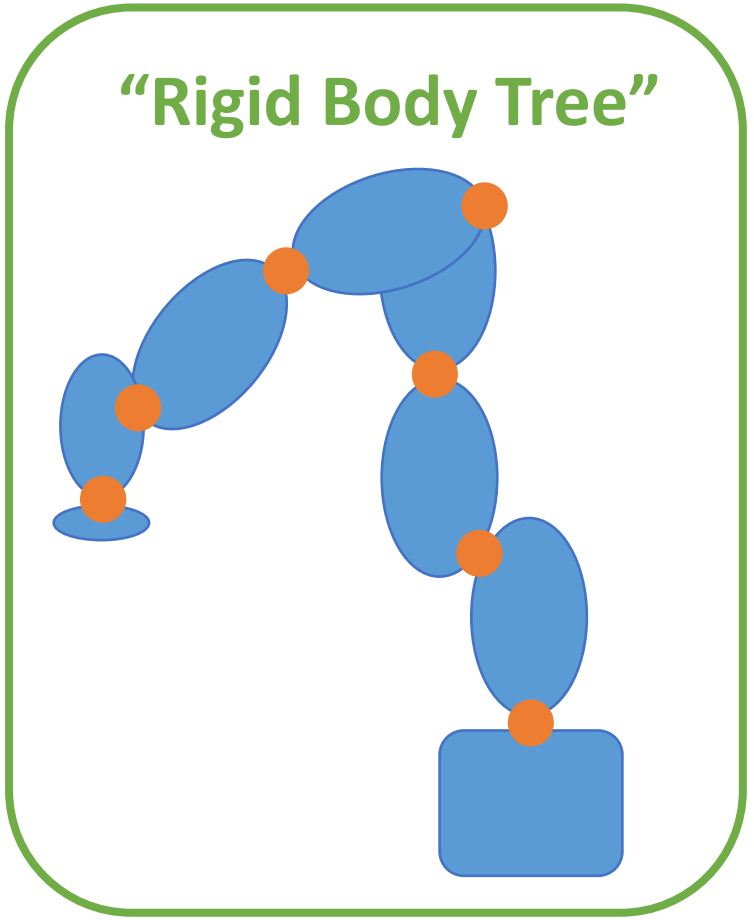
What is Rigid Body Dynamics?



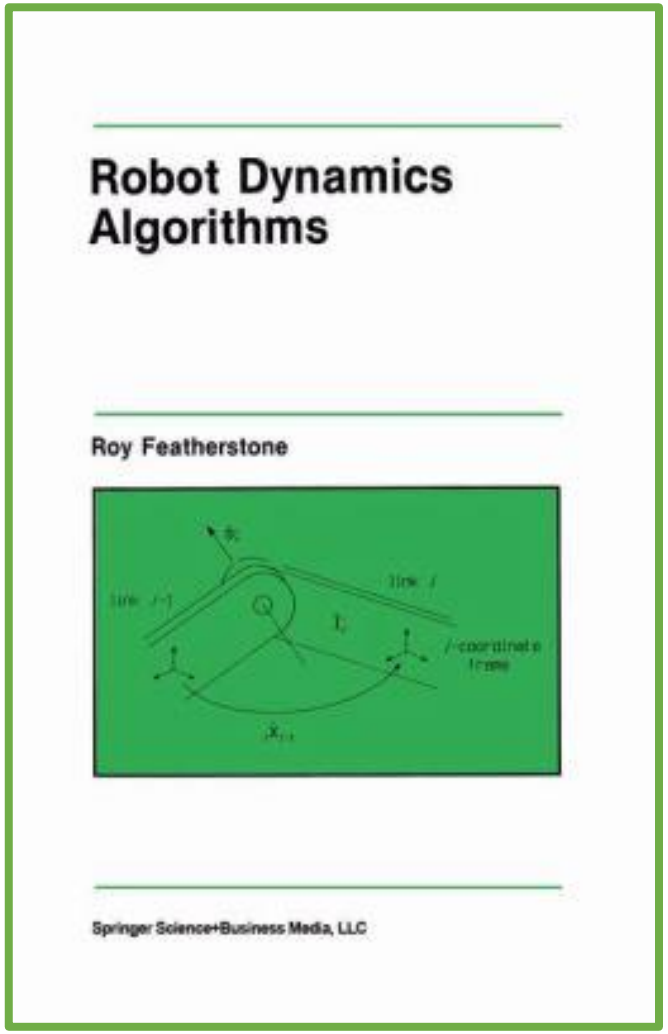
“Rigid Body Tree”

What is Rigid Body Dynamics?

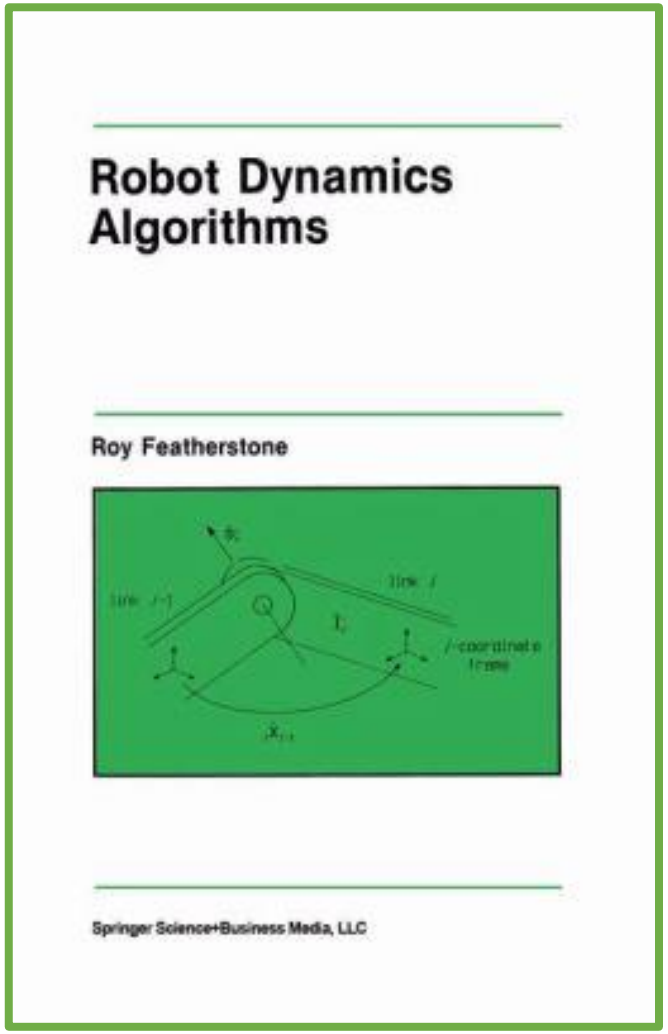




What is Rigid Body Dynamics?



What is Rigid Body Dynamics?



The study of the motion of bodies under the action of forces **(aka $F = ma$)**

How do Rigid Body Dynamics Algorithms Work?

How do Rigid Body Dynamics Algorithms Work?

Algorithm 1 RNEA$(q, \dot{q}, \ddot{q}, f^{ext}) \rightarrow c$

2: **for** link $i = 1 : n$ **do**

Joints/Links are numbered in order

7: **for** link $i = n : 1$ **do**

The algorithms

up and down the tree

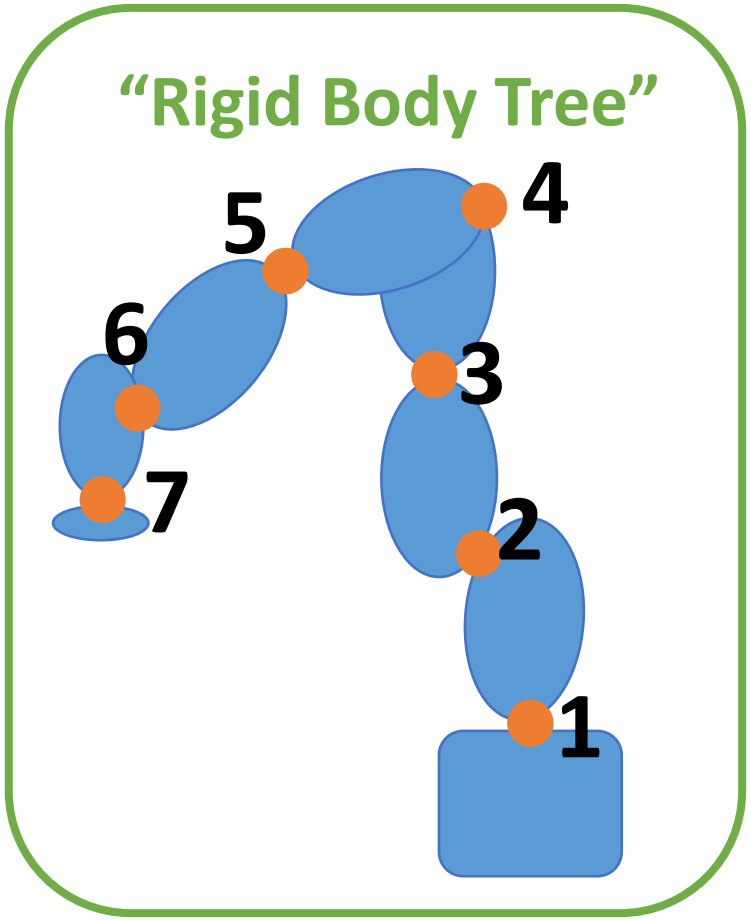
How do Rigid Body Dynamics Algorithms Work?

Algorithm 1 RNEA$(q, \dot{q}, \ddot{q}, f^{ext}) \rightarrow c$

2: **for** link $i = 1 : n$ **do**

7: **for** link $i = n : 1$ **do**

Joints/Links are numbered in order



How do Rigid Body Dynamics Algorithms Work?

Algorithm 1 RNEA$(q, \dot{q}, \ddot{q}, f^{ext}) \rightarrow c$

2: **for** link $i = 1 : n$ **do**

4: $v_i =$

5: $a_i =$

6: $f_i =$

7: **for** link $i = n : 1$ **do**

8: c_i

9: f_{λ_i}

λ refers to the parent link

The algorithms

pass the **forces (f)**, **velocities (v)**, and **accelerations (a)** **up and down the tree**

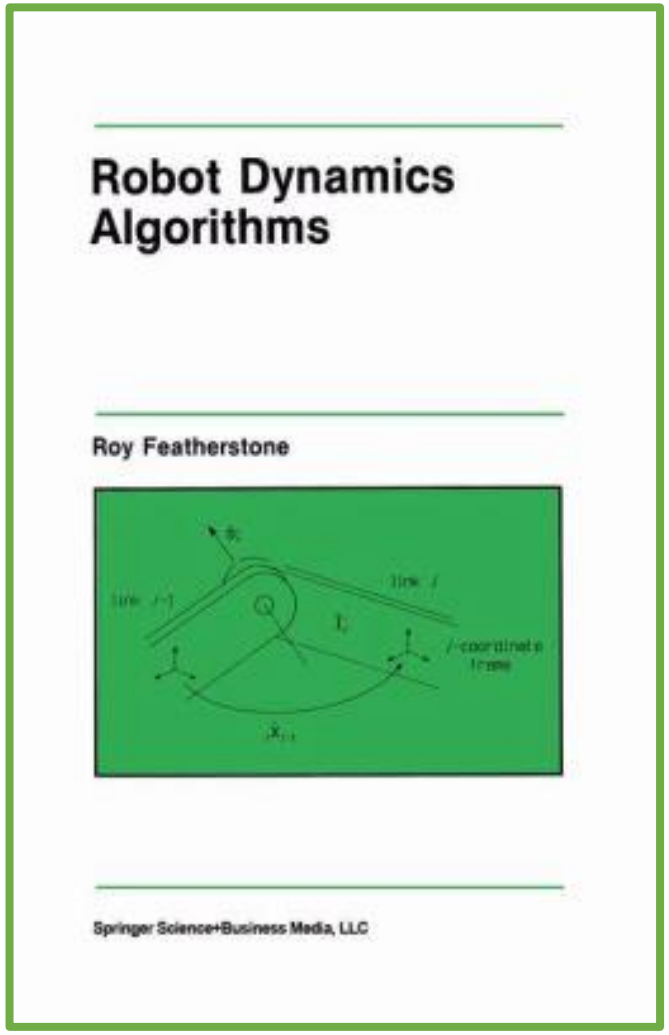
How do Rigid Body Dynamics Algorithms Work?

Algorithm 1 RNEA$(q, \dot{q}, \ddot{q}, f^{ext}) \rightarrow c$

1: $v_0 = 0,\ a_0 = \text{gravity}$
2: **for** link $i = 1 : n$ **do**
3: Compute ${}^{i}X_{\lambda_i}, S_i, I_i$
4: $v_i = {}^{i}X_{\lambda_i} v_{\lambda_i} + S_i \dot{q}_i$
5: $a_i = {}^{i}X_{\lambda_i} a_{\lambda_i} + S_i \ddot{q}_i + v_i \times S_i \dot{q}_i$
6: $f_i = I_i a_i + v_i \times^* I_i v_i - f_i^{\text{ext}}$
7: **for** link $i = n : 1$ **do**
8: $c_i = S_i^T f_i$
9: $f_{\lambda_i} \mathrel{+}= {}^{i}X_{\lambda_i}^T f_i$

The algorithms use **transformations (X)** and **inertias (I)** along with **joint direction vectors (S)** and **cross products ($\times$)** to pass the **forces (f)**, **velocities (v)**, and **accelerations (a)** **up and down the tree**

Rigid Body Dynamics Algorithms



- The study of the motion of bodies under the action of forces $(F = ma)$
- It **loops up and down** the rigid body tree
- It uses a **variety of different operations** in each step

What is the Gradient of Rigid Body Dynamics?

The multi-dimensional derivative

Algorithm 2 $\nabla\text{RNEA}(\dot{q}, v, a, f, X, S, I)$ $\partial c/\partial u$

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

2: $$\frac{\partial v_i}{\partial u} = {}^{i}X_{\lambda_i}\frac{\partial v_{\lambda_i}}{\partial u} + \begin{cases} ({}^{i}X_{\lambda_i} v_{\lambda_i}) \times S_i & u = q \\ S_i & u \equiv \dot{q} \end{cases}$$

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

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

5: **for** link $i = N : 1$ **do**

6: $$\frac{\partial c_i}{\partial u} = S_i^T\frac{\partial f_i}{\partial u}$$

7: $$\frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^{i}X_{\lambda_i}^T\frac{\partial f_i}{\partial u} + {}^{i}X_{\lambda_i}^T(S_i \times^* f_i)$$

Basically the same algorithm (just a lot of *dus*)

Algorithmic Features of the Gradient of Rigid Body Dynamics as a step of a planning and control algorithm

Algorithmic Feature	CPU	GPU
Parallelism	Bad	Good
Serial Dependencies	Good	
Irregular Operations		

Each of the cores is designed to be good at general purpose computations

Algorithmic Features of the Gradient of Rigid Body Dynamics as a step of a planning and control algorithm

Algorithmic Feature	CPU	GPU
Parallelism	Bad	Good
Serial Dependencies	Good	Bad
Irregular Operations		

All cores are optimized for regular and parallel operations

Algorithmic refactoring is needed to effectively target GPUs

Algorithm 2 $\nabla\text{RNEA}(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

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

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

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

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

5: **for** link $i = N : 1$ **do**

6: $$\frac{\partial c_i}{\partial u} = S_i^T\frac{\partial f_i}{\partial u}$$

7: $$\frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^{i}X_{\lambda_i}^T\frac{\partial f_i}{\partial u} + {}^{i}X_{\lambda_i}^T\left(S_i \times^* f_i\right)$$

Algorithmic refactoring is needed to effectively target GPUs

Generate temporary values in parallel to reduce the computations done in serial loops

Algorithm 2 $\nabla\text{RNEA}(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

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

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

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

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

5: **for** link $i = N : 1$ **do**

6: $$\frac{\partial c_i}{\partial u} = S_i^T\frac{\partial f_i}{\partial u}$$

7: $$\frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^iX_{\lambda_i}^T\frac{\partial f_i}{\partial u} + {}^iX_{\lambda_i}^T(S_i \times^* f_i)$$

Algorithm 4 $\nabla\text{RNEA-GPU}(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

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

2: $\alpha_i = {}^iX_{\lambda_i}v_{\lambda_i}$ $\beta_i = {}^iX_{\lambda_i}a_{\lambda_i}$ $\gamma_i = I_iv_i$

3: $\alpha_i = \alpha_i \times S_i$ $\beta_i = \beta_i \times S_i$ $\delta_i = v_i \times S_i$
$\zeta_i = f_i \times S_i$ $\eta_i = v_i\times^*$

4: $\zeta_i = -{}^iX_{\lambda_i}^T\zeta_i$ $\eta_i = \eta_iI_i$

5: **for** link $i = 1 : n$ **do**

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

7: **for** link $i = 1 : n$ **in parallel do**

8: $$\mu_i = \frac{\partial v_i}{\partial u}\times^* \quad \rho_i = \frac{\partial v_{\lambda_i}}{\partial u} \times S_i\dot{q}_i + \begin{cases} \beta_i \\ \delta_i \end{cases}$$

9: **for** link $i = 1 : n$ **do**

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

11: **for** link $i = 1 : n$ **in parallel do**

12: $$\frac{\partial f_i}{\partial u} = I_i\frac{\partial a_i}{\partial u} + \mu_i\gamma_i + \eta_i\frac{\partial v_i}{\partial u}$$

13: **for** link $i = n : 1$ **do**

14: $$\frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^iX_{\lambda_i}^T\frac{\partial f_i}{\partial u} + \zeta_i$$

15: **for** link $i = n : 1$ **in parallel do**

16: $$\frac{\partial c_i}{\partial u} = S_i^T\frac{\partial f_i}{\partial u}$$

Algorithmic refactoring is needed to effectively target GPUs

Re-ordering computations to unify operations

Algorithm 2 $\nabla\text{RNEA}(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

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

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

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

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

5: **for** link $i = N : 1$ **do**

6: $$\frac{\partial c_i}{\partial u} = S_i^T\frac{\partial f_i}{\partial u}$$

7: $$\frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^iX_{\lambda_i}^T\frac{\partial f_i}{\partial u} + {}^iX_{\lambda_i}^T(S_i \times^* f_i)$$

Algorithm 4 $\nabla\text{RNEA-GPU}(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

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

2: $\alpha_i = {}^iX_{\lambda_i}v_{\lambda_i}$ $\beta_i = {}^iX_{\lambda_i}a_{\lambda_i}$ $\gamma_i = I_iv_i$

3: $\alpha_i = \alpha_i \times S_i$ $\beta_i = \beta_i \times S_i$ $\delta_i = v_i \times S_i$

$\zeta_i = f_i \times S_i$ $\eta_i = v_i\times^*$

4: $\zeta_i = -{}^iX_{\lambda_i}^T\zeta_i$ $\eta_i = \eta_iI_i$

5: **for** link $i = 1 : n$ **do**

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

7: **for** link $i = 1 : n$ **in parallel do**

8: $$\mu_i = \frac{\partial v_i}{\partial u} \times^* \quad \rho_i = \frac{\partial v_{\lambda_i}}{\partial u} \times S_i\dot{q}_i + \begin{cases} \beta_i \\ \delta_i \end{cases}$$

9: **for** link $i = 1 : n$ **do**

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

11: **for** link $i = 1 : n$ **in parallel do**

12: $$\frac{\partial f_i}{\partial u} = I_i\frac{\partial a_i}{\partial u} + \mu_i\gamma_i + \eta_i\frac{\partial v_i}{\partial u}$$

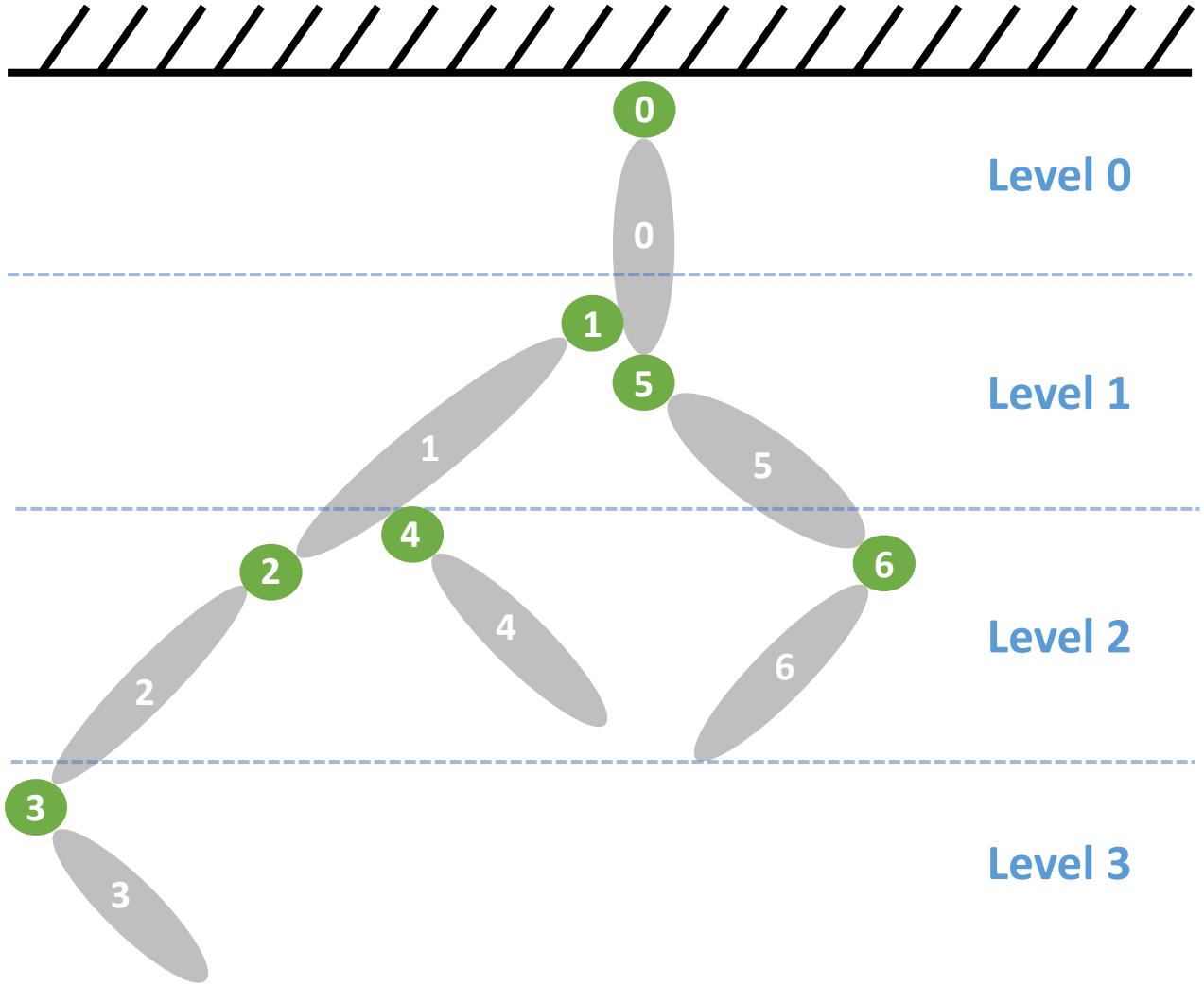
13: **for** link $i = n : 1$ **do**

14: $$\frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^iX_{\lambda_i}^T\frac{\partial f_i}{\partial u} + \zeta_i$$

15: **for** link $i = n : 1$ **in parallel do**

16: $$\frac{\partial c_i}{\partial u} = S_i^T\frac{\partial f_i}{\partial u}$$

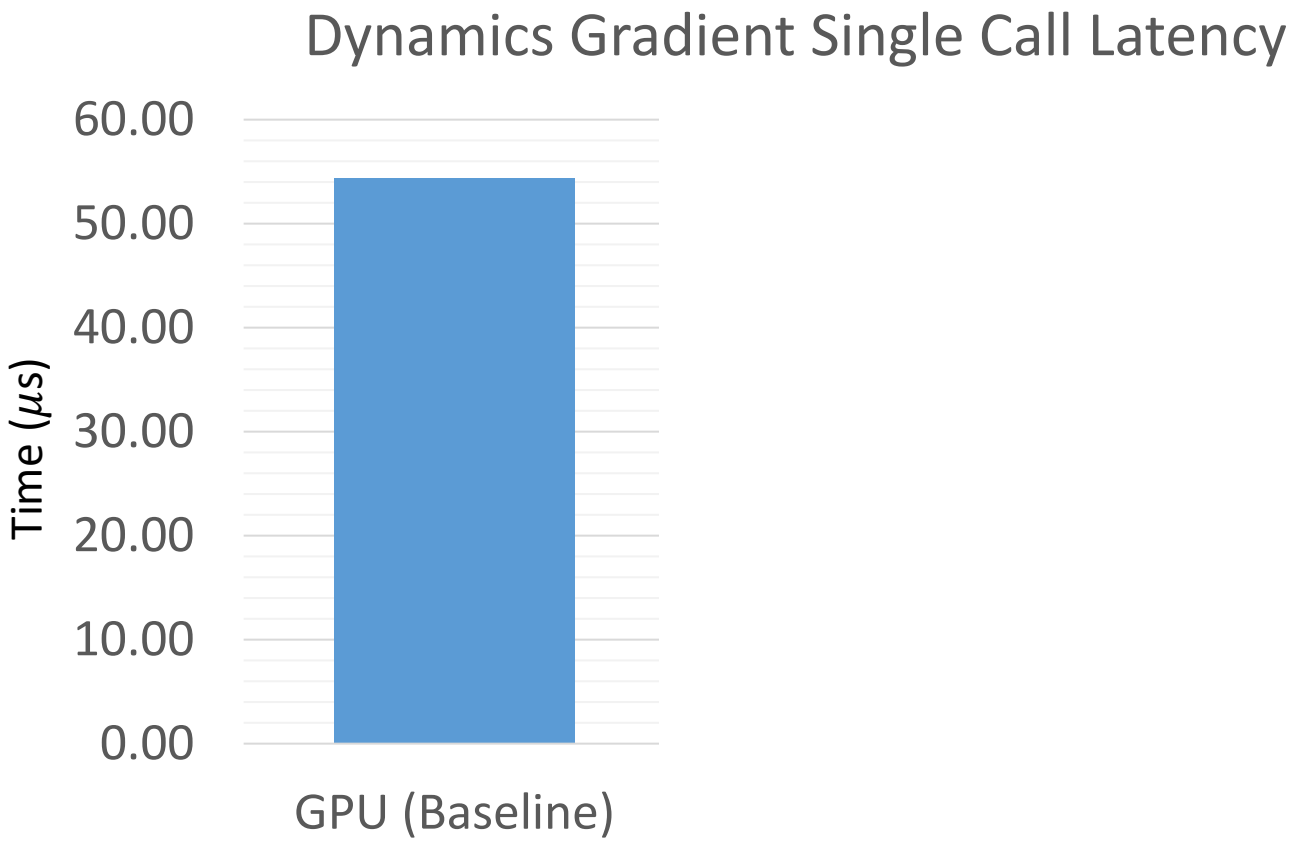
Algorithmic refactoring is needed to effectively target GPUs



Algorithm 4 ∇RNEA-GPU$(\dot{q}, v, a, f, X, S, I) \rightarrow \partial c/\partial u$

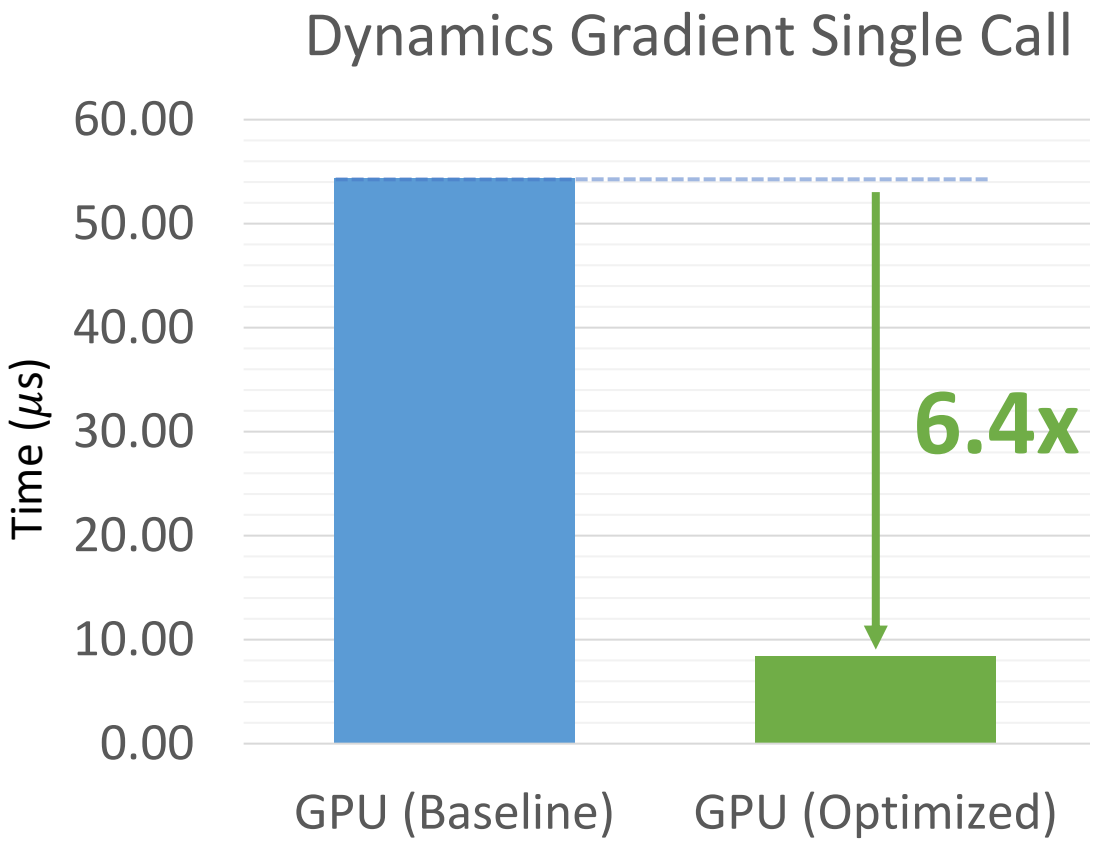
1: **for** link $i = 1 : n$ **in parallel do**
2: $\quad \alpha_i = {}^iX_{\lambda_i} v_{\lambda_i} \quad \beta_i = {}^iX_{\lambda_i} a_{\lambda_i} \quad \gamma_i = I_i v_i$
3: $\quad \alpha_i = \alpha_i \times S_i \quad \beta_i = \beta_i \times S_i \quad \delta_i = v_i \times S_i$
$\quad \zeta_i = f_i \times S_i \quad \eta_i = v_i \times^*$
4: $\quad \zeta_i = -{}^iX_{\lambda_i}^T \zeta_i \quad \eta_i = \eta_i I_i$
5: **for** link $i = 1 : n$ **do** — In parallel within each level
6: $\quad \frac{\partial v_i}{\partial u} = {}^iX_{\lambda_i} \frac{\partial v_{\lambda_i}}{\partial u} + \begin{cases} \alpha_i & u = q \\ S_i & u \equiv \dot{q} \end{cases}$
7: **for** link $i = 1 : n$ **in parallel do**
8: $\quad \mu_i = \frac{\partial v_i}{\partial u} \times^* \quad \rho_i = \frac{\partial v_{\lambda_i}}{\partial u} \times S_i \dot{q}_i + \begin{cases} \beta_i \\ \delta_i \end{cases}$
9: **for** link $i = 1 : n$ **do** — In parallel within each level
10: $\quad \frac{\partial a_i}{\partial u} = {}^iX_{\lambda_i} \frac{\partial a_{\lambda_i}}{\partial u} + \rho_i$
11: **for** link $i = 1 : n$ **in parallel do**
12: $\quad \frac{\partial f_i}{\partial u} = I_i \frac{\partial a_i}{\partial u} + \mu_i \gamma_i + \eta_i \frac{\partial v_i}{\partial u}$
13: **for** link $i = n : 1$ **do** — In parallel within each level
14: $\quad \frac{\partial f_{\lambda_i}}{\partial u} \mathrel{+}= {}^iX_{\lambda_i}^T \frac{\partial f_i}{\partial u} + \zeta_i$
15: **for** link $i = n : 1$ **in parallel do**
16: $\quad \frac{\partial c_i}{\partial u} = S_i^T \frac{\partial f_i}{\partial u}$

Refactoring improved single computation latency



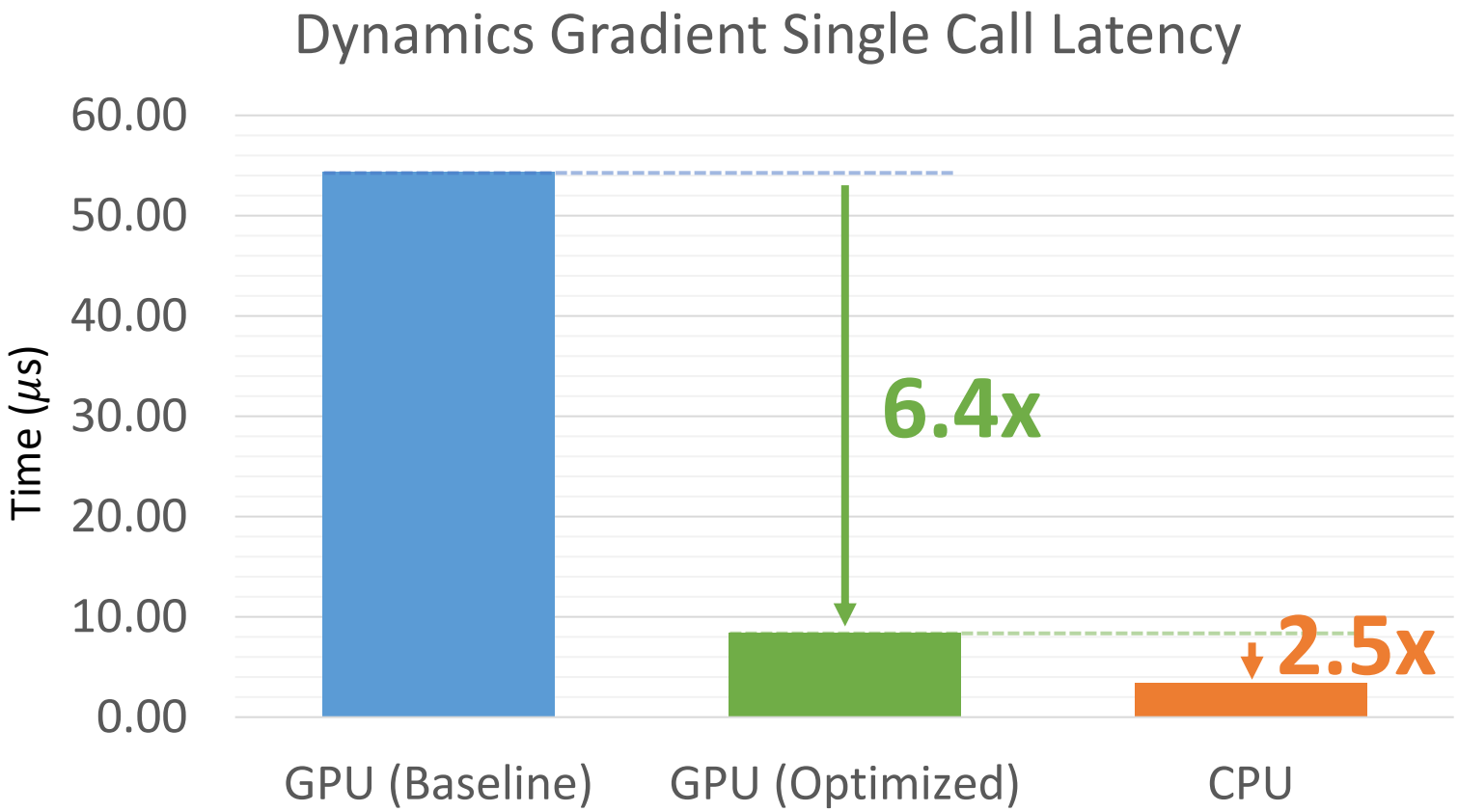
B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

Refactoring improved single computation latency



B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

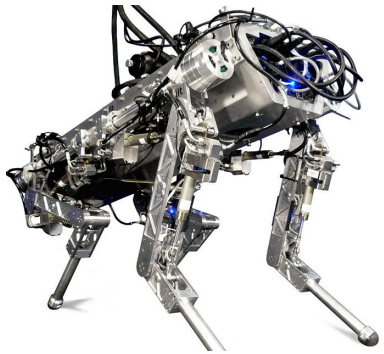
Refactoring improved single computation latency



B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

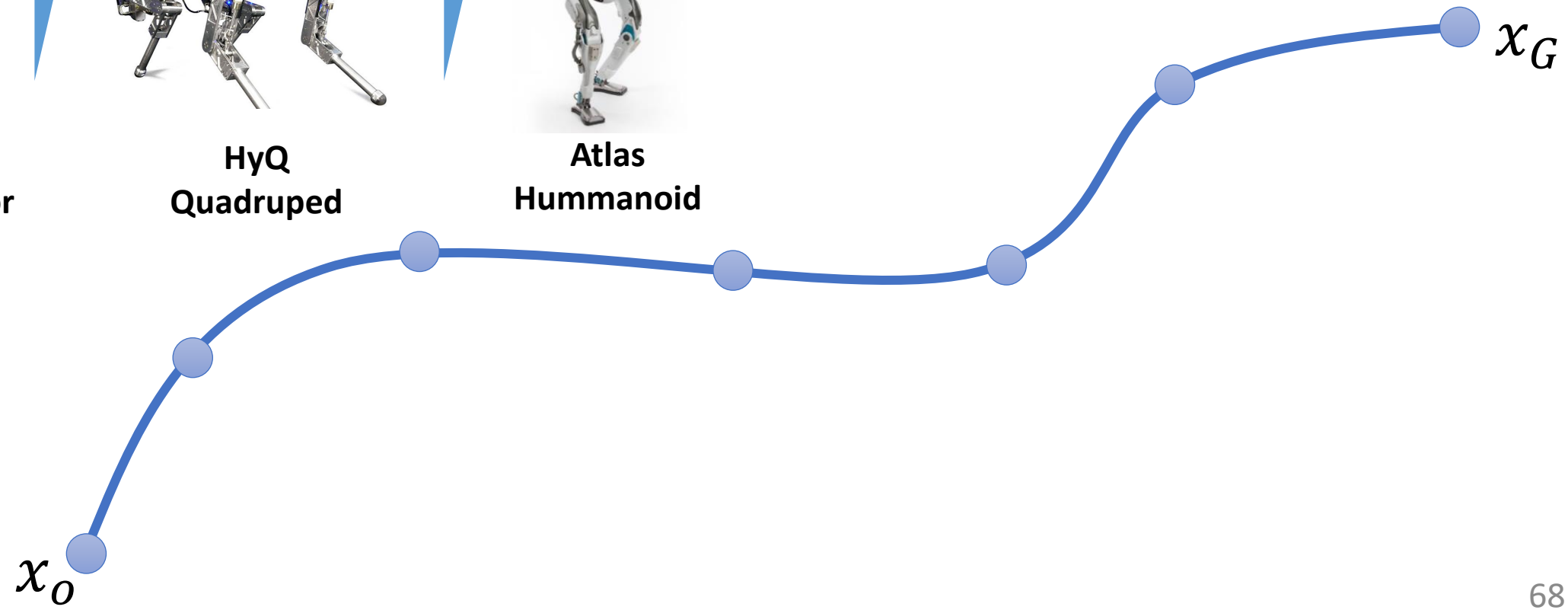
What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?

Kuka IIWA Manipulator

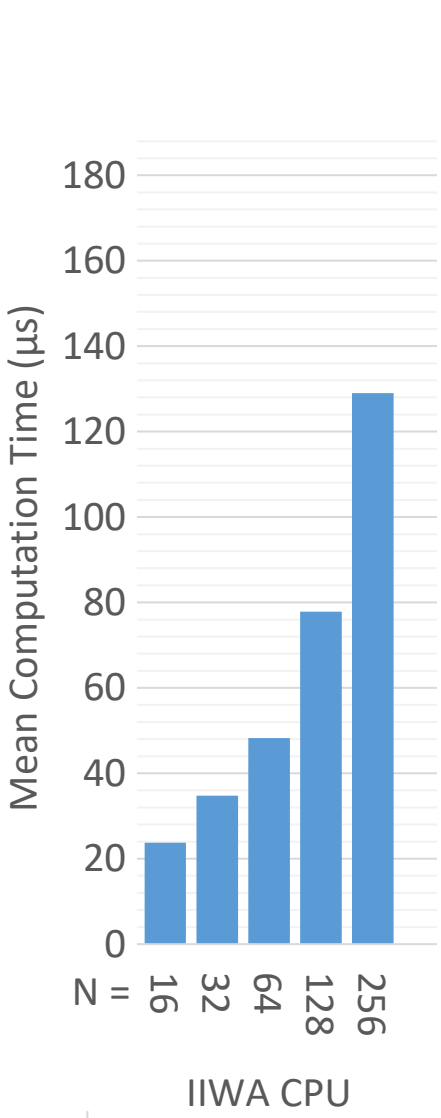


HyQ Quadruped

Atlas Hummanoid

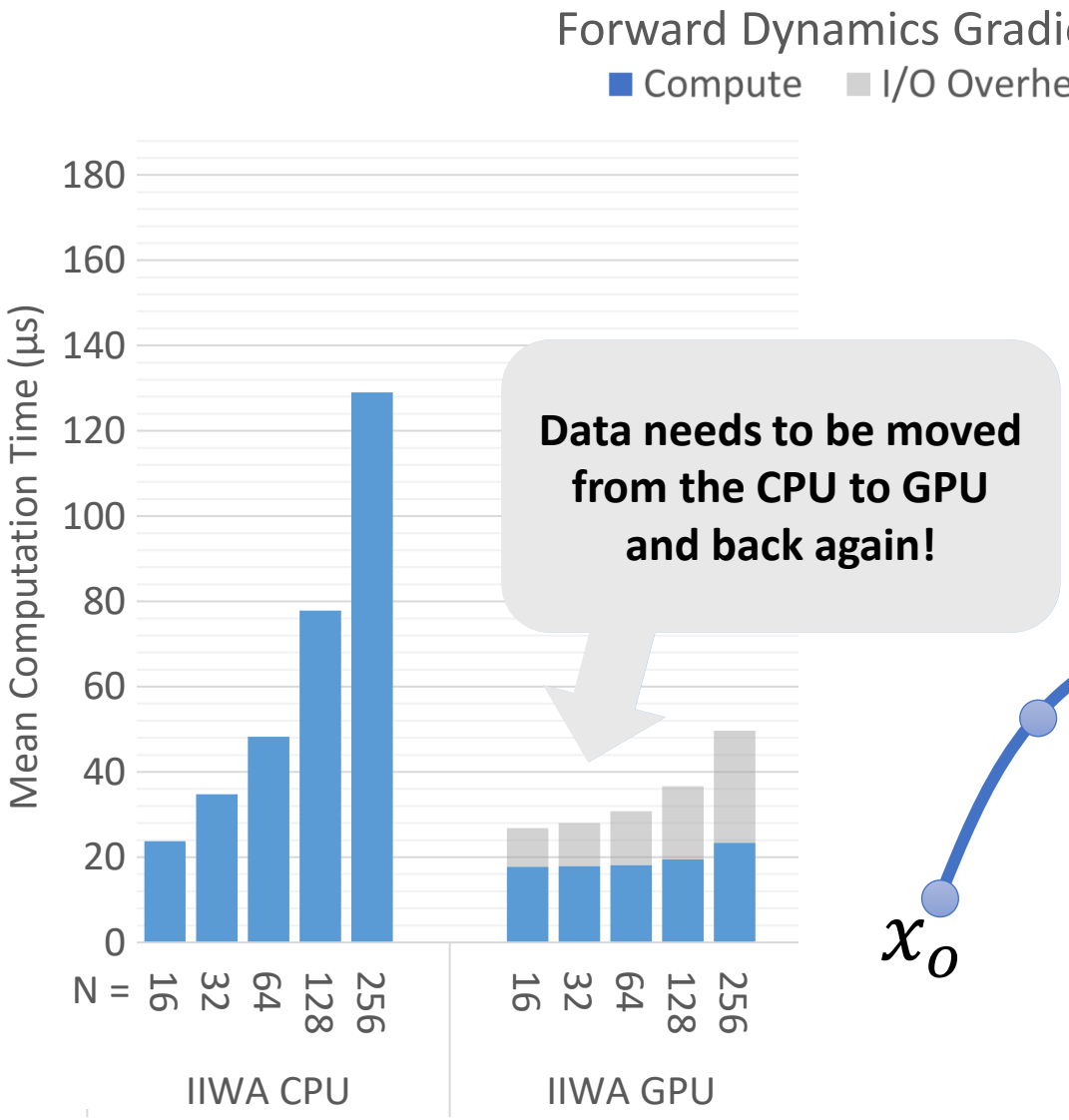


What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



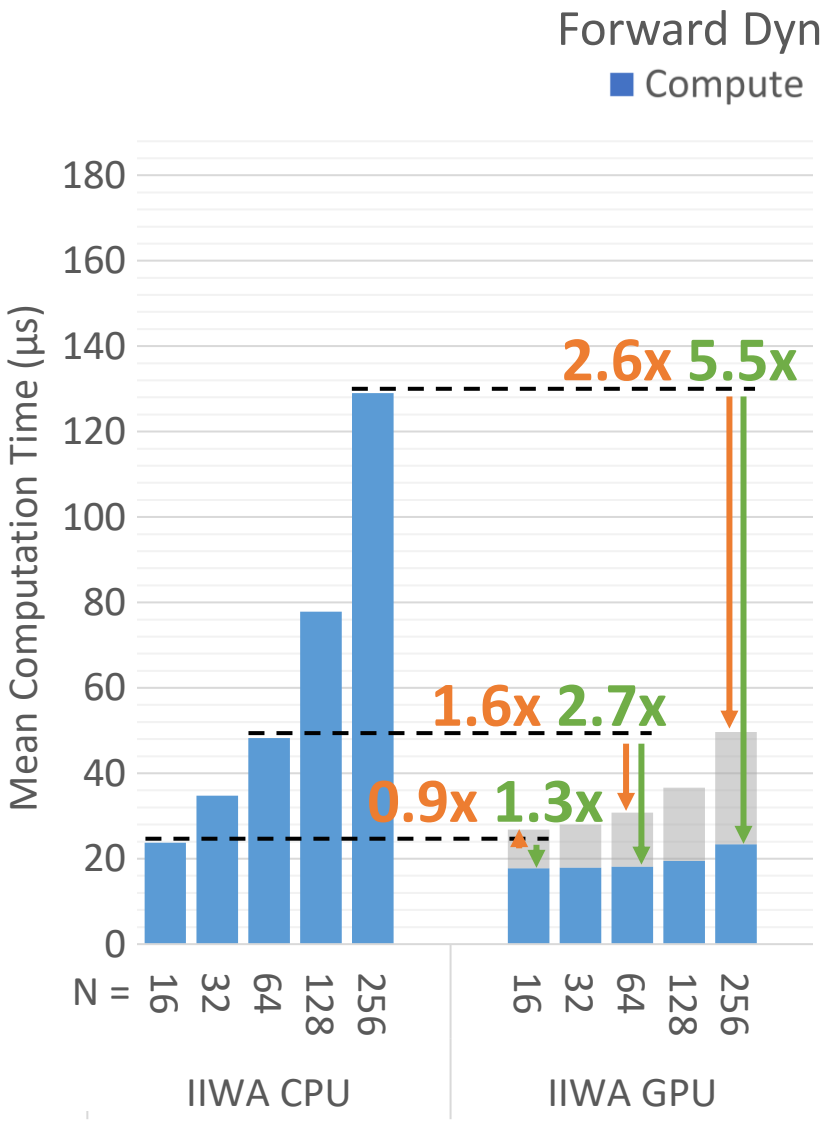
[Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.

What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



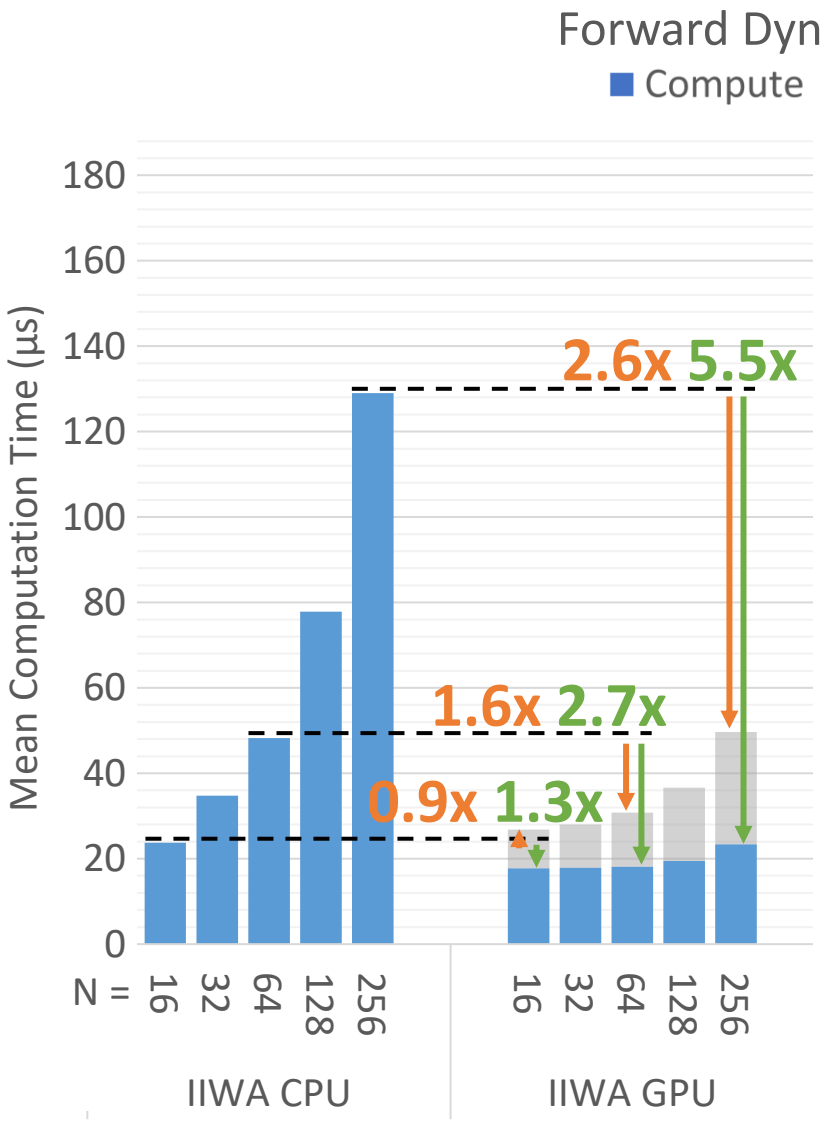
[Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.

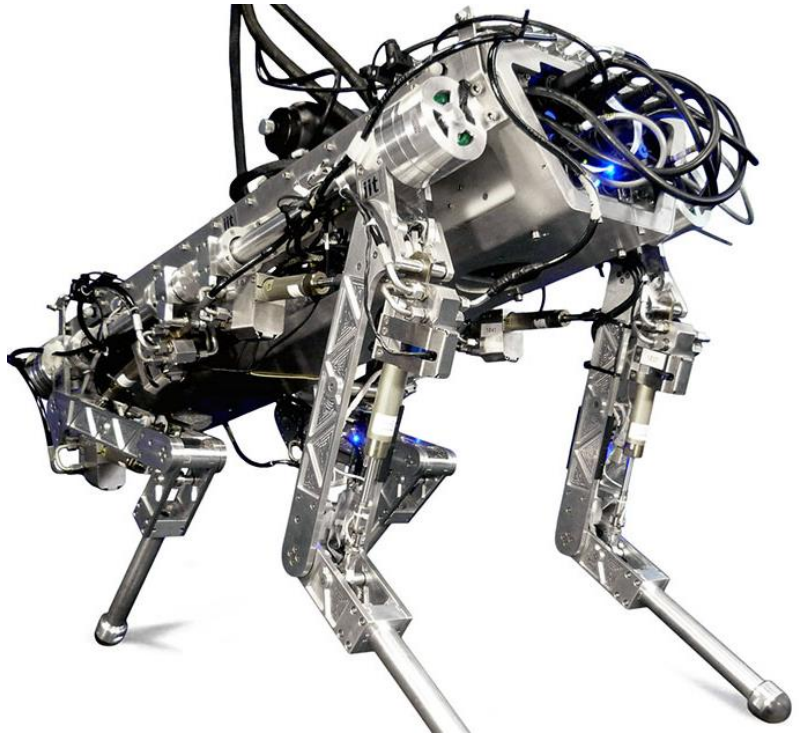
What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



[Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.

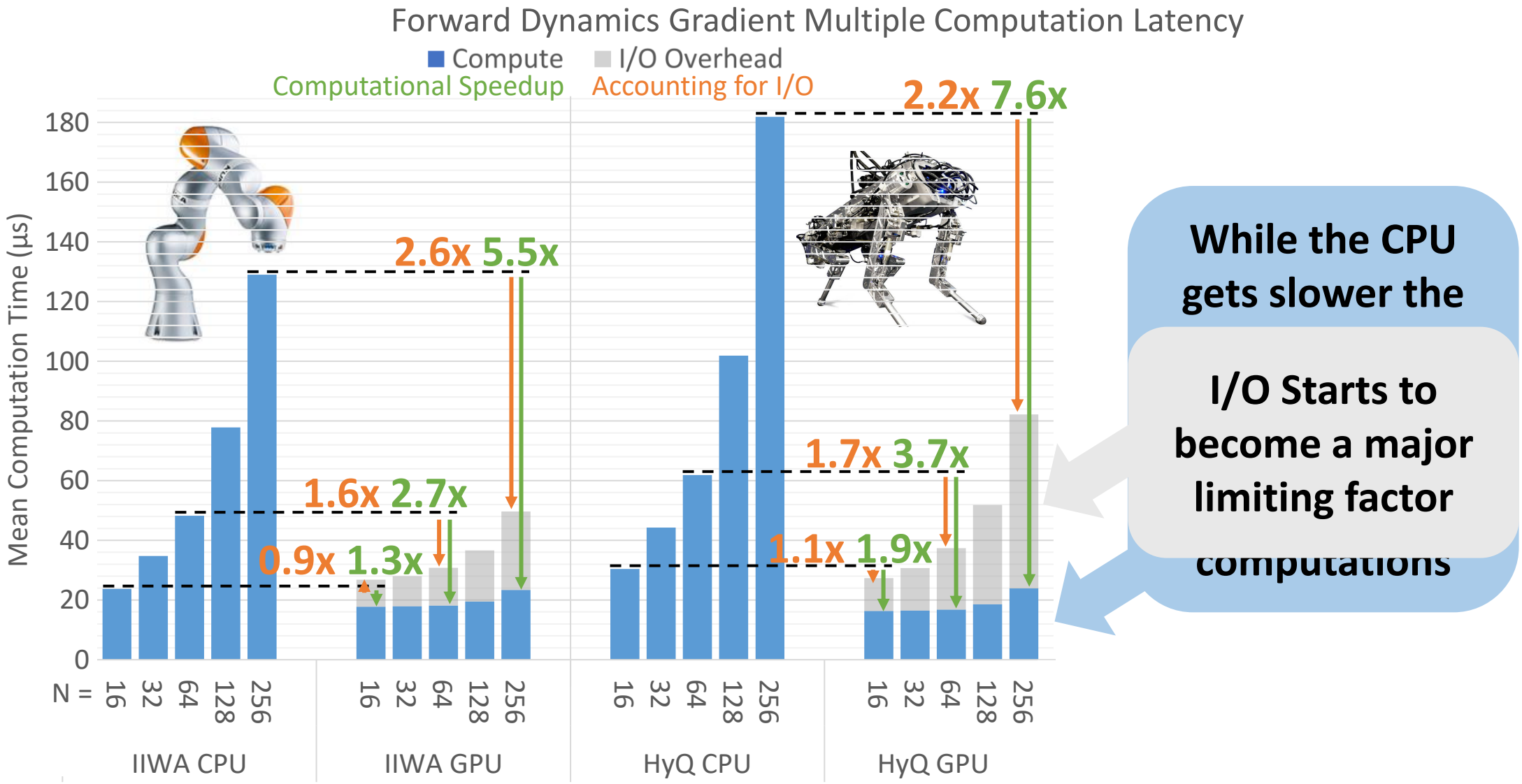
What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



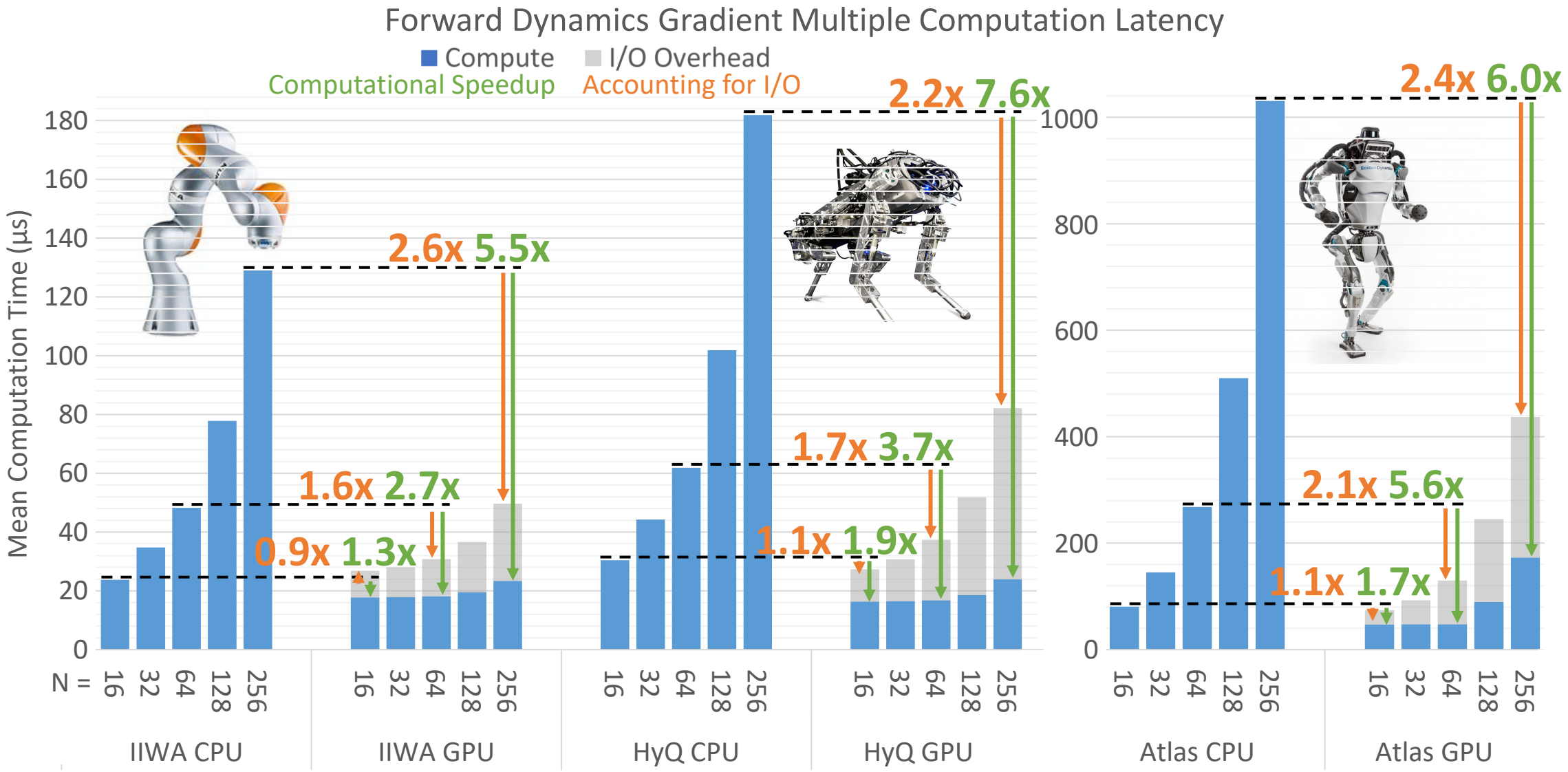


[Under Review] **B. Plancher**, S. M. Neuman, R. Ghosal, S. Kuindersma, V. Janapa Reddi. GRiD: GPU-Accelerated Rigid Body Dynamics with Analytical Gradients. ICRA, 2022.

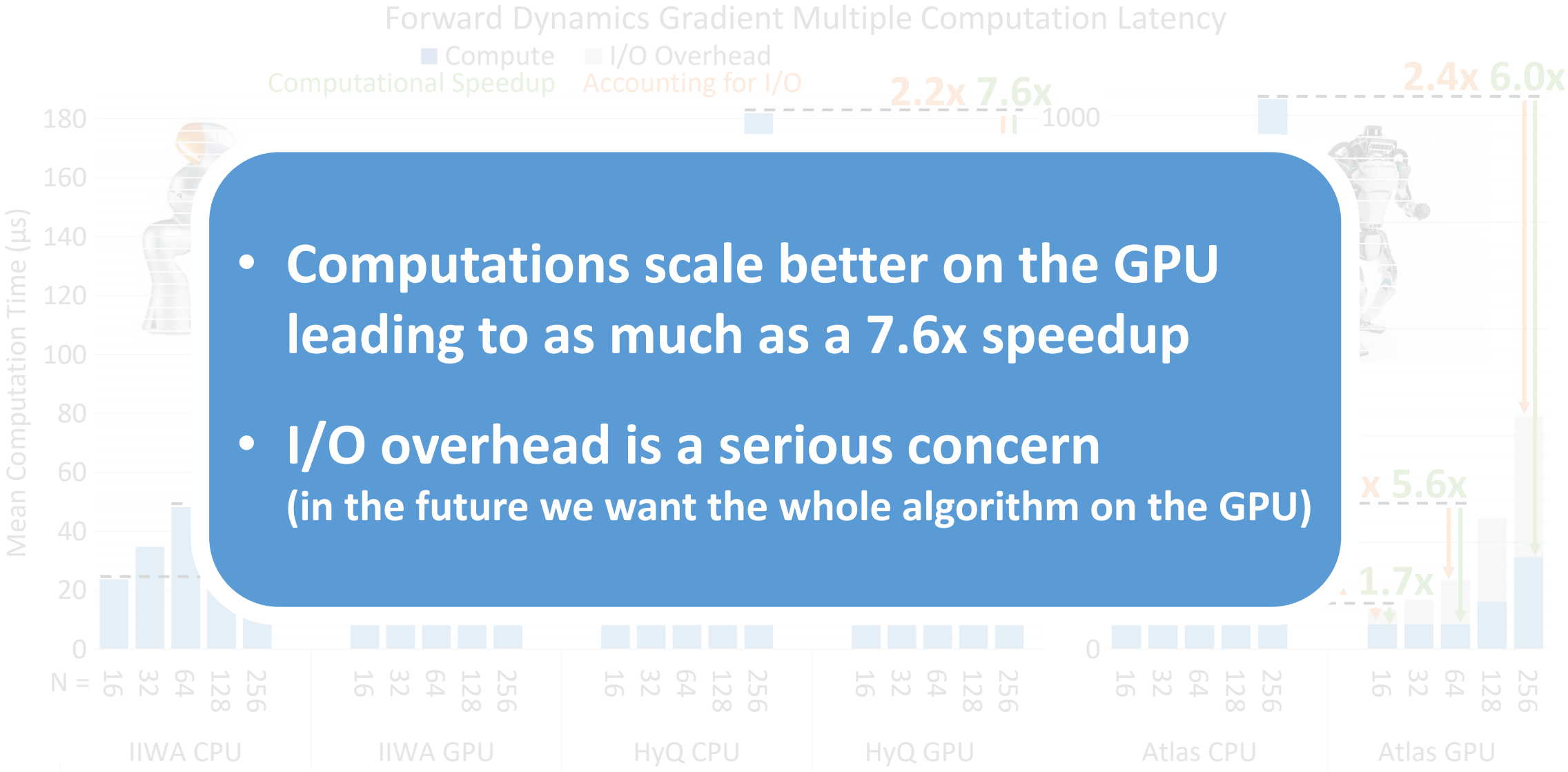
What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



What happens when we Scale Robot Complexity and consider Multiple Simultaneous Computations?



- **Computations scale better on the GPU leading to as much as a 7.6x speedup**
- **I/O overhead is a serious concern**
  (in the future we want the whole algorithm on the GPU)

Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

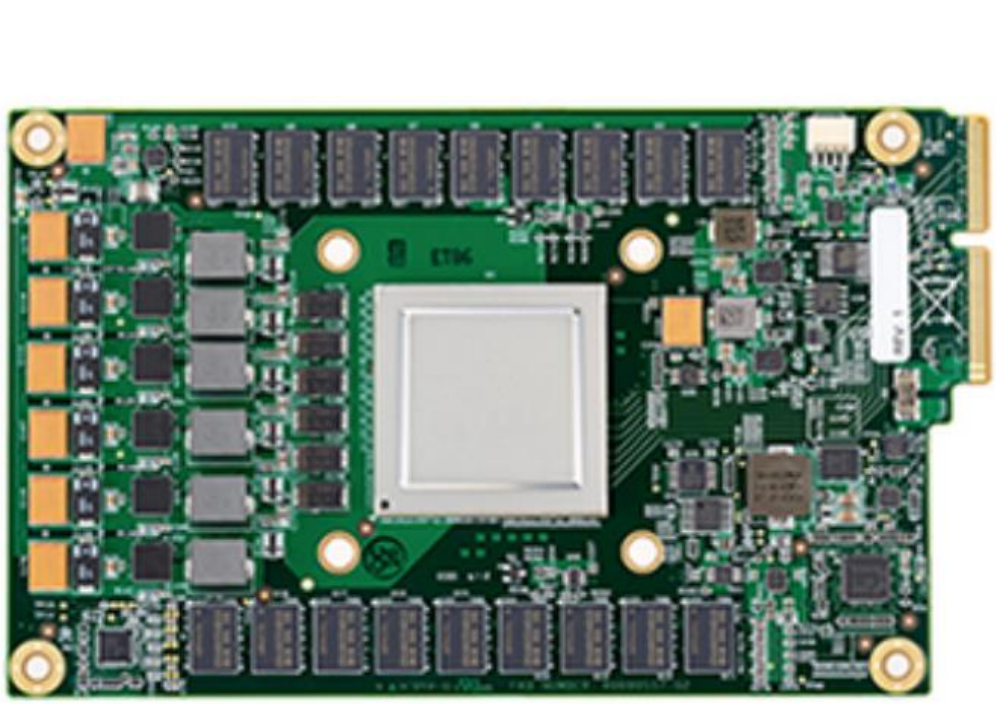
A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

- Why Accelerate Rigid Body Dynamics?
- Accelerating Rigid Body Dynamics on the GPU
- **Moving Beyond the GPU**

The Future of Accelerating Robotics

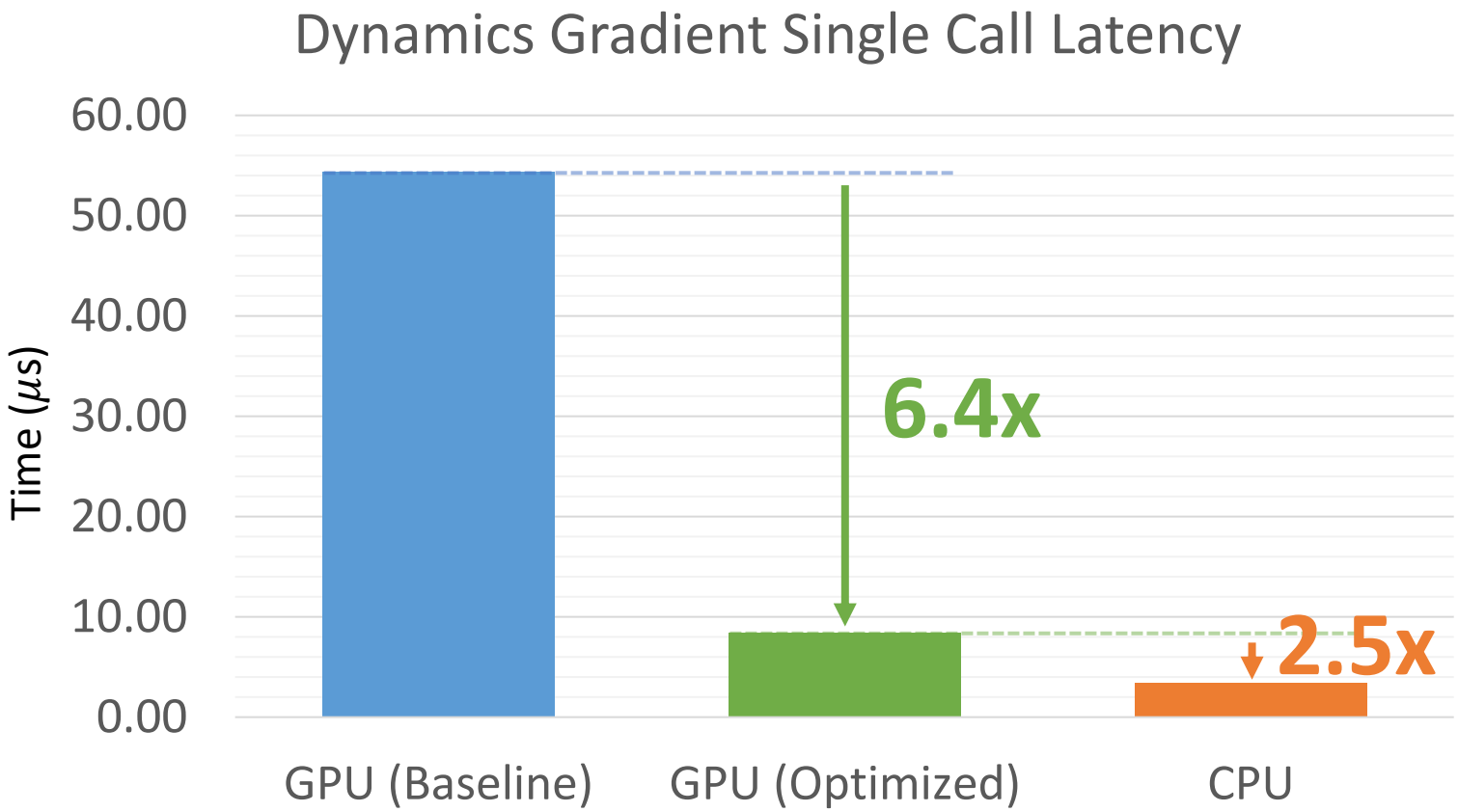
Custom Hardware Can Provide More Acceleration



Google's first Tensor Processing Unit (TPU) on a printed circuit board (left); TPUs deployed in a Google datacenter (right)

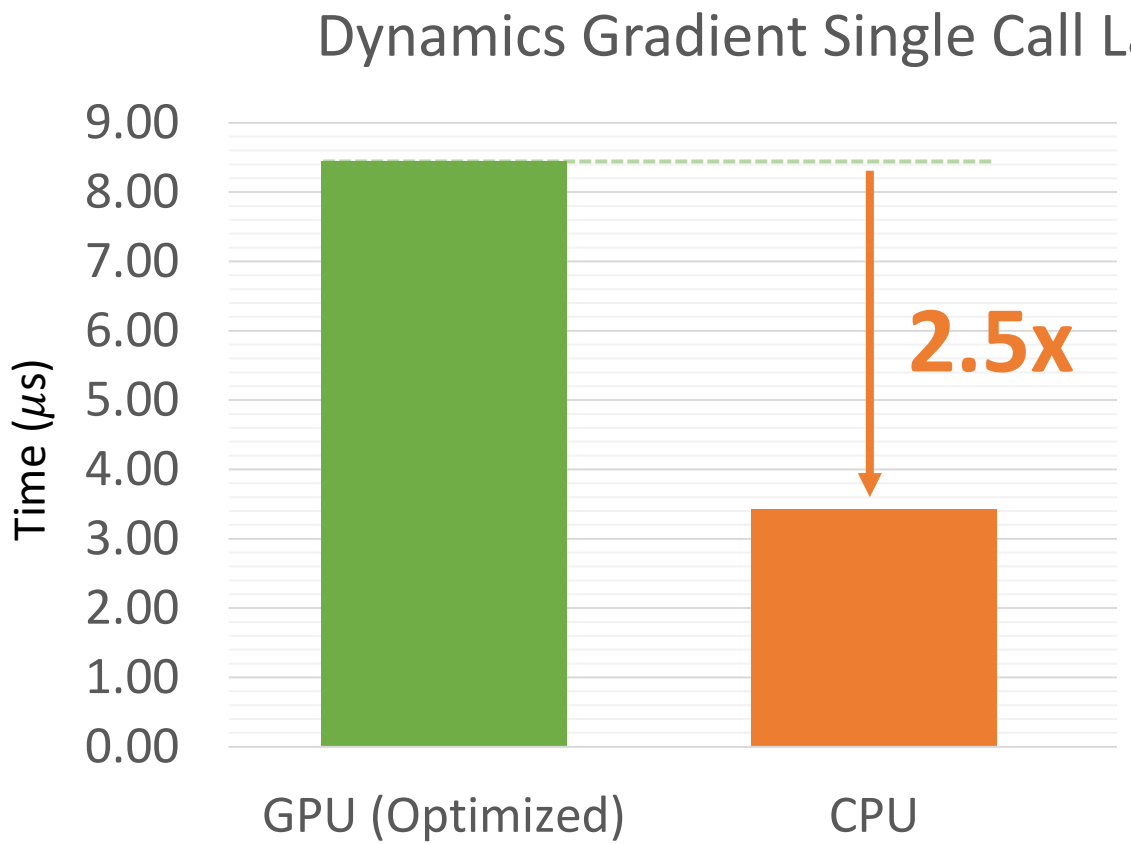
https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

Custom Hardware Can Provide More Acceleration



B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

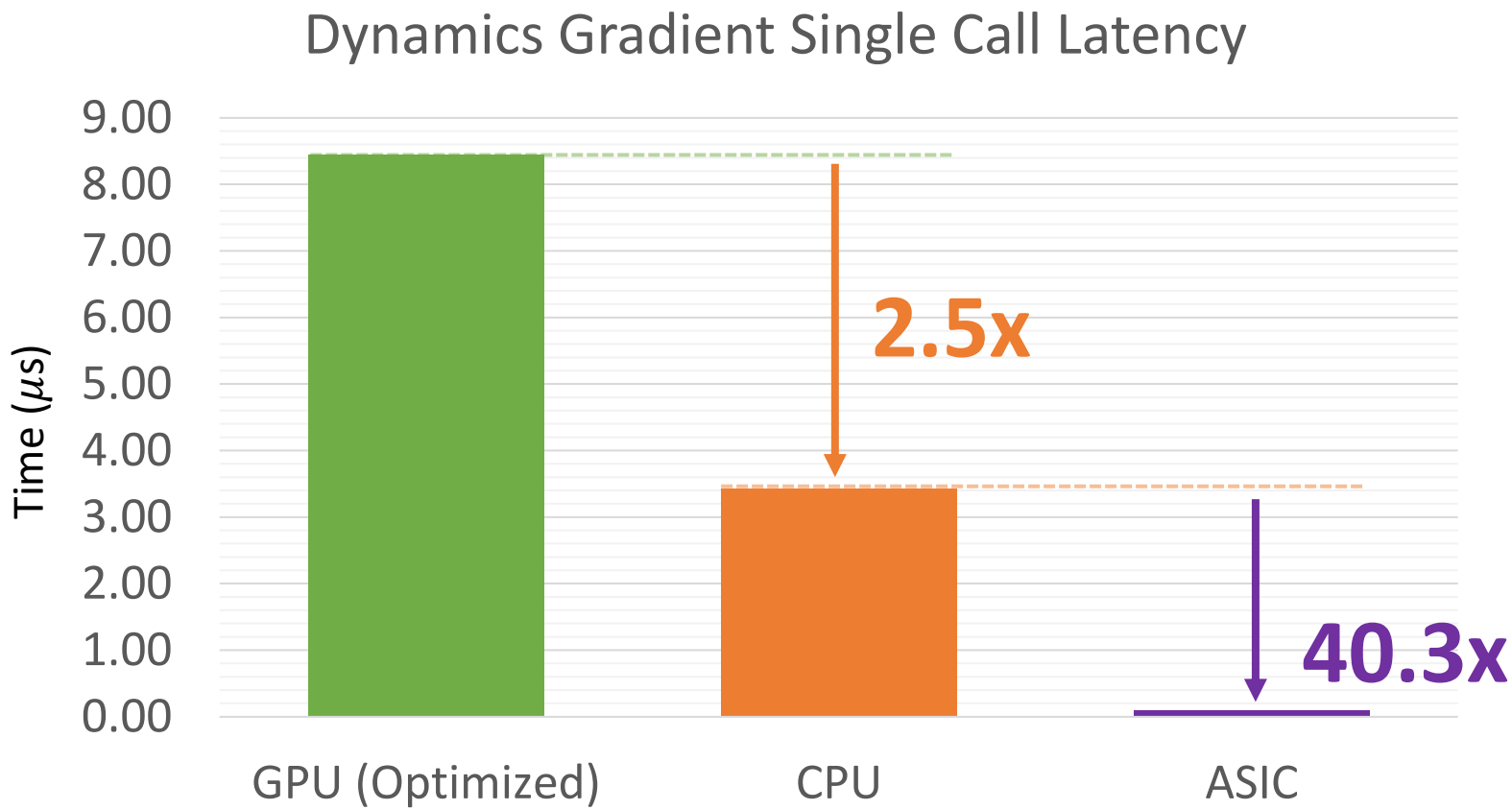
Custom Hardware Can Provide More Acceleration



B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

S. M. Neuman, **B. Plancher**, T. Bourgeat, T. Tambe, S. Devadas, V. Janapa Reddi. Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology. ASPLOS 2021.

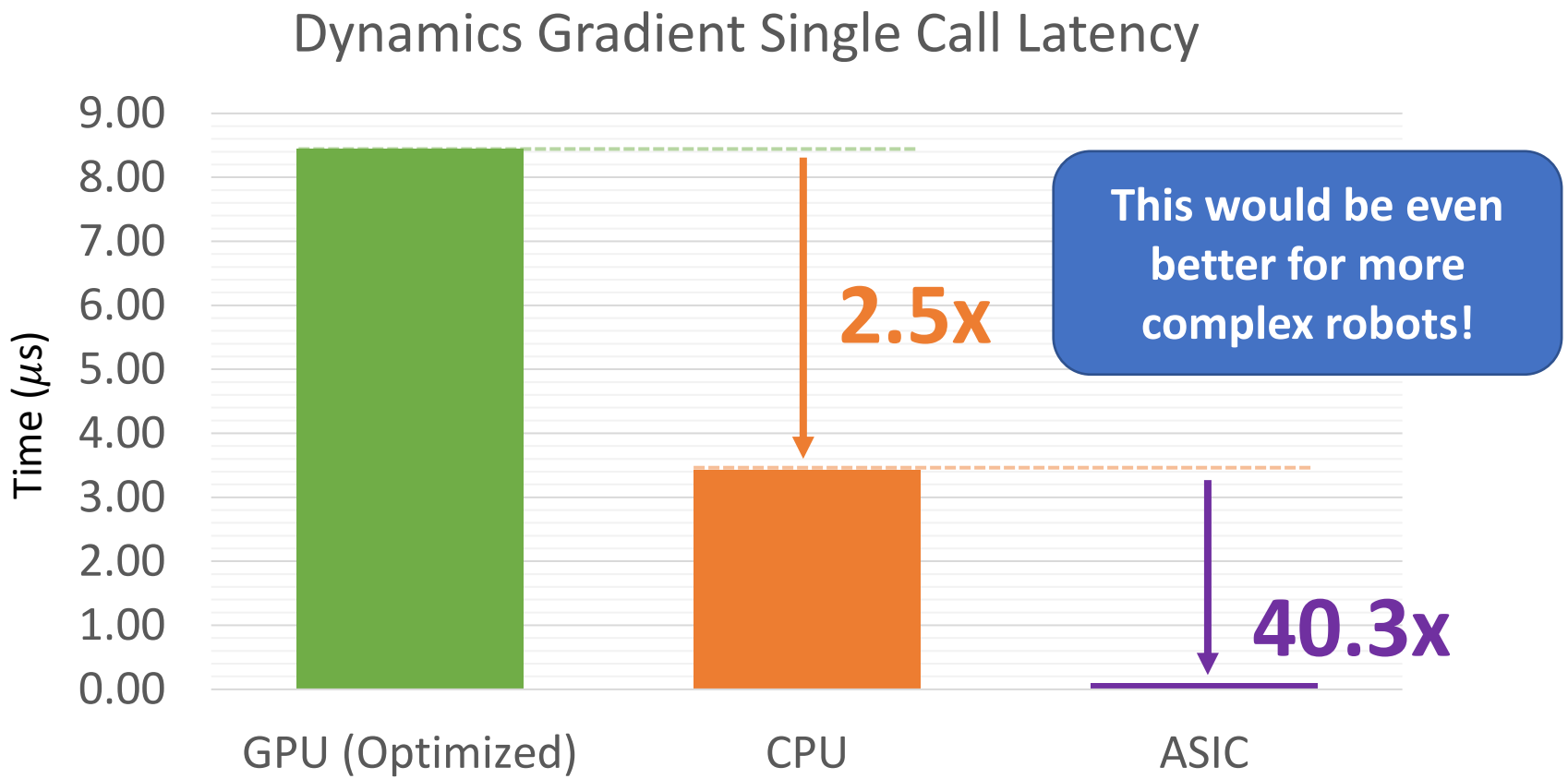
Custom Hardware Can Provide More Acceleration



B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

S. M. Neuman, **B. Plancher**, T. Bourgeat, T. Tambe, S. Devadas, V. Janapa Reddi. Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology. ASPLOS 2021.

Custom Hardware Can Provide More Acceleration



B. Plancher, S. M. Neuman, T. Bourgeat, S. Kuindersma, S. Devadas, V. Janapa Reddi. Accelerating Robot Dynamics Gradients on a CPU, GPU, and FPGA. RA-L and ICRA 2021.

S. M. Neuman, **B. Plancher**, T. Bourgeat, T. Tambe, S. Devadas, V. Janapa Reddi. Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology. ASPLOS 2021.

Hardware Acceleration for Realtime Robotics

A Quick Overview of (And My Approach To) Robotics

A Crash Course in (GPU) Architecture

A Case Study in Accelerating Rigid Body Dynamics

The Future of Accelerating Robotics

Enabling Robotics Research on High-Performance Parallel Architectures

CONTROL TOOLBOX

ROPIC

acados – a modular open-source framework for fast embedded optimal control

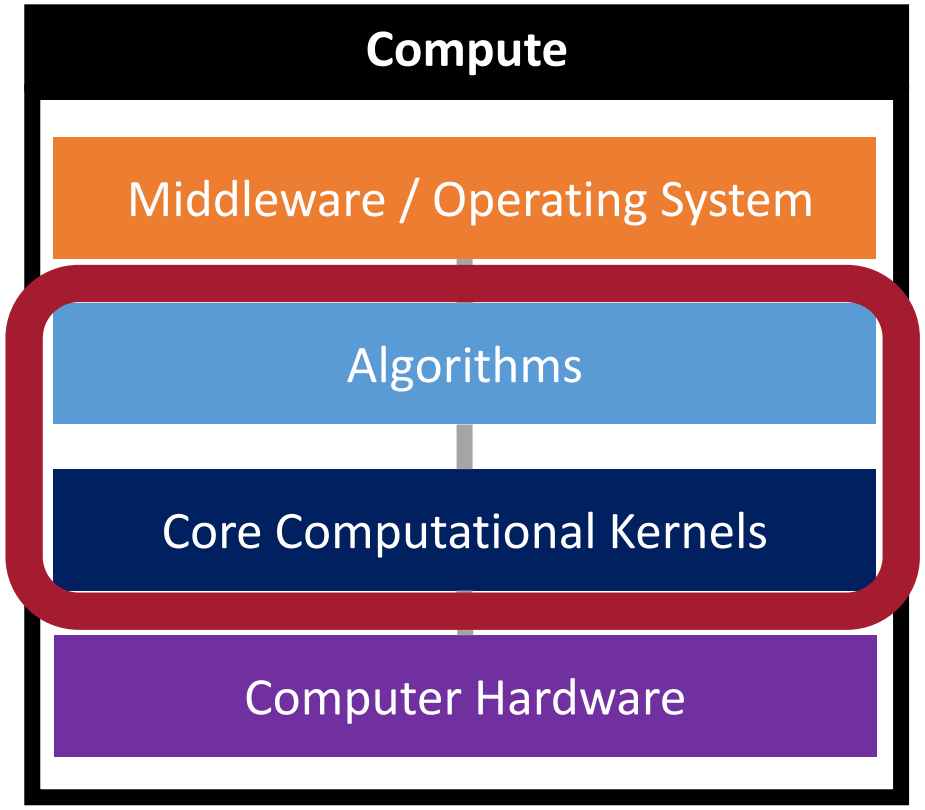
Only Forward Dynamics On GPU

NVIDIA PHYSX

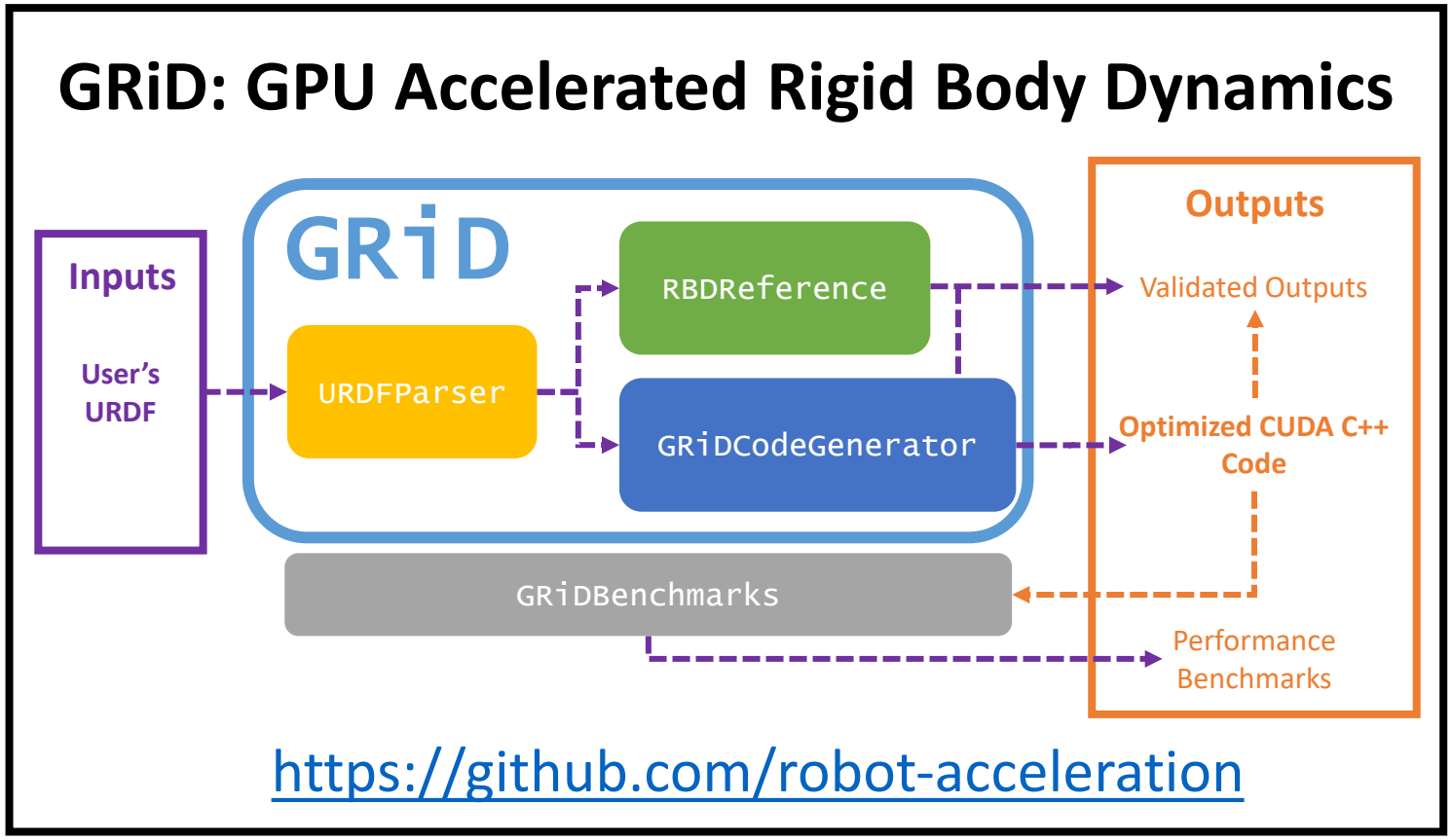
DRAKE

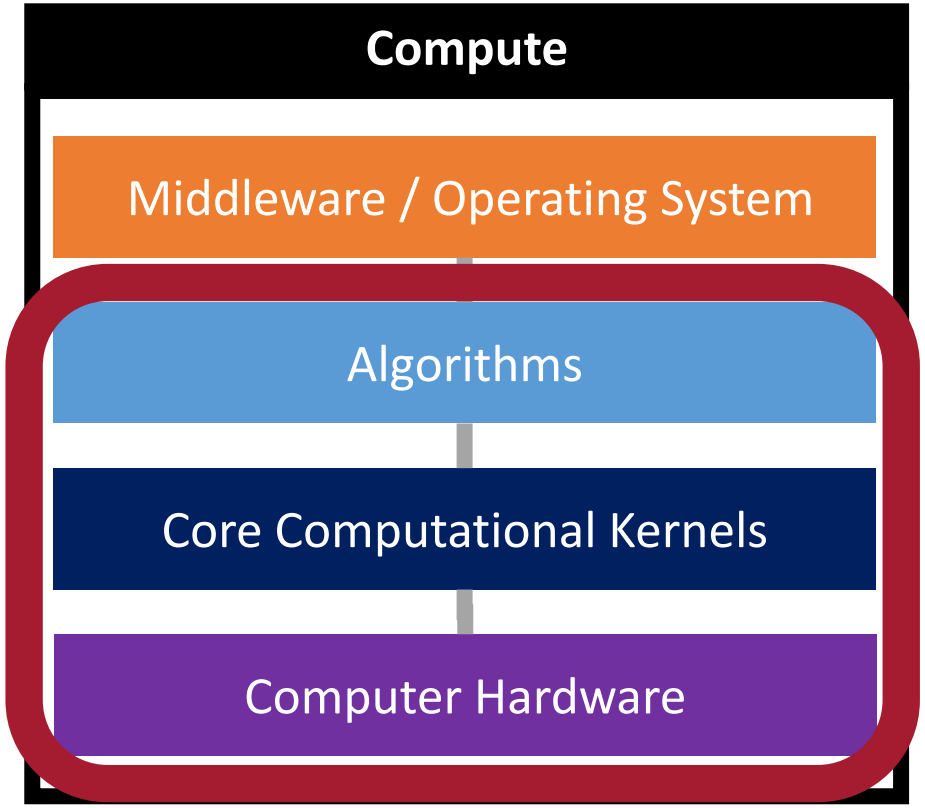
MODEL-BASED DESIGN AND VERIFICATION FOR ROBOTICS

CPU ONLY!

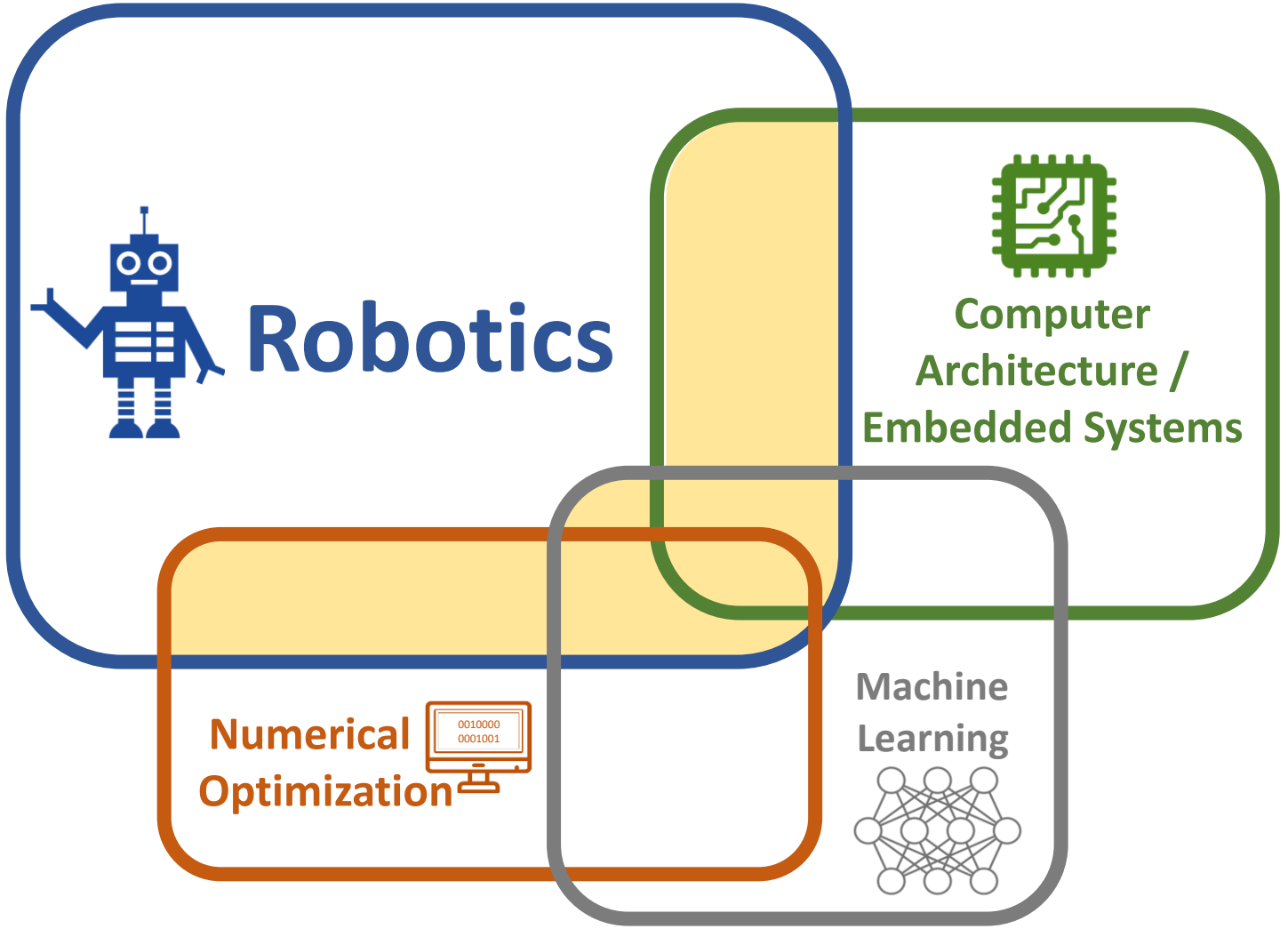


Enabling Robotics Research on High-Performance Parallel Architectures

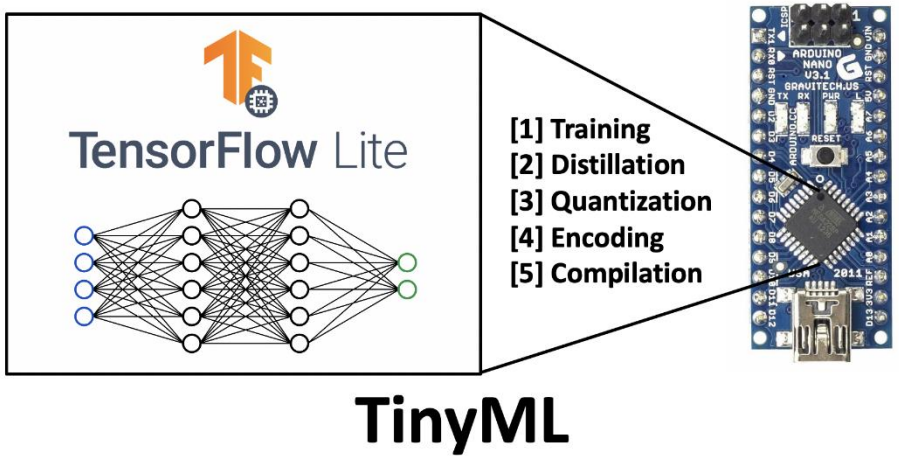




I work at the intersection of robotics and adjacent fields



Tiny Robotics: Lowering the Cost of Robotics for Widespread Deployment



Thank You to My Funders and Collaborators

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), and the Defense Advanced Research Projects Agency (under Grant HR001118C0018). 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

Hardware Acceleration for Realtime Robotics

https://brianplancher.com brian_plancher@g.harvard.edu

1. Learn some of the **language of robotics**
   (and computer architecture)
2. Understand the important of **parallelism**
3. Gain practice in exploring the **process** and **opportunities** of **hardware acceleration**

Feedback Link: https://bit.ly/Brian-Barnard-21