Realtime Model Predictive Control using Parallel DDP on a GPU

Brian Plancher and Scott Kuindersma

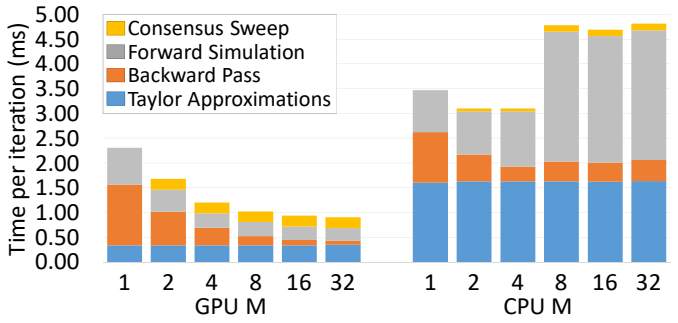
The Big Picture:

We analyze the benefits and tradeoffs of parallelization by comparing a GPU and multi-threaded CPU implementation of a multiple-shooting variant of Differential Dynamic Programming (DDP) using a Kuka LBR IIWA-14 manipulator. Our results suggest that GPU-based solvers can offer faster per-iteration computation time and faster convergence in some cases, but in general tradeoffs exist between convergence behavior and degree of algorithm-level parallelism. We also find that in an online Model Predictive Control (MPC) setting this approach is robust to the presence of model discrepancies and communication delays. Finally, we find that higher control rates generally lead to better tracking across a range of parallelization options.

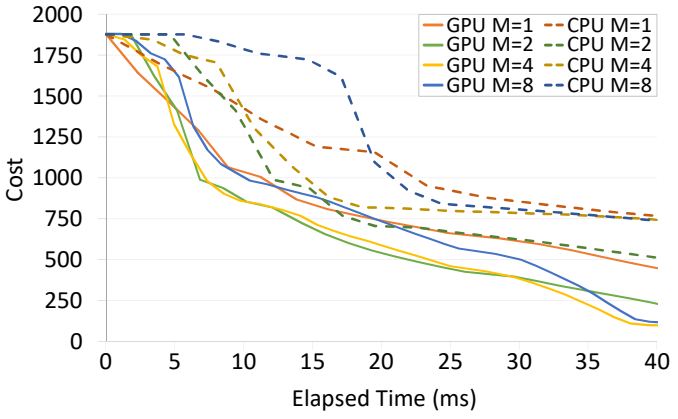
Simulation Results:

We found that parallelism:

- Increased the speed of the Taylor approximations
- Had diminishing returns for the forward simulation and backward pass
- Was only effective while there were available cores
- Decreased the convergence rate leading to non-monotonic forward simulation times on the CPU



These tradeoffs led to the GPU outperforming the CPU across parallelization options



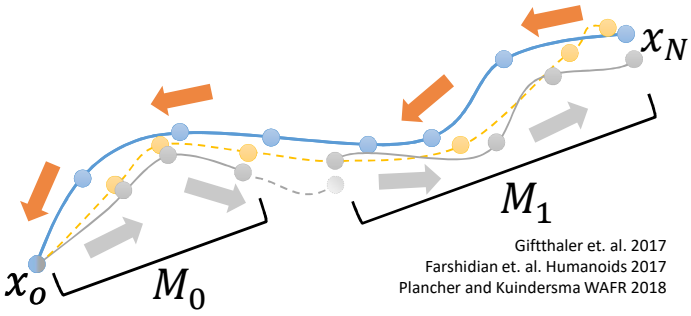
This work was supported by a Draper Internal Research and Development grant and by the National Science Foundation Graduate Research Fellowship (under grant DGE1745303). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the funding organizations.

HARVARD
John A. Paulson
School of Engineering
and Applied Sciences

Parallel DDP:

Parallel DDP takes the classic DDP algorithm and induces **algorithm level parallelism** by replacing the**:**

- Serial backward pass with an M block parallel solve
- Serial forward pass with a fast serial consensus sweep using linearized dynamics followed by an M leg multiple shooting forward simulation.



Parallel DDP uses **instruction level parallelism** to compute in parallel the:

- Taylor approximations of the cost and dynamics
- A set of possible line search iterates

Hardware Results:

We ran an online MPC figure-eight goal tracking experiment and found that:

- This approach was robust to model discrepancies and communication delays
- Faster control steps led to reduced tracking error
- Solvers started to fail when they had about as many (or less) iterations as the amount of algorithm level parallelism (e.g., M = 4 with 3 or less iterations)