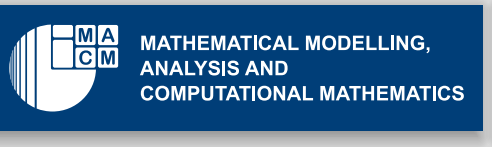
Task graph-based performance analysis of PinT methods

Jens Hahne[1]
In collaboration with: Stephanie Friedhoff[1] and Matthias Bolten[2]

[1]Bergische Universität Wuppertal, Germany



Introduction

- Goals:
  - Performance model for PinT methods
  - Static load balacing
  - Dynamic load balancing

Introduction

- Goals:
  - Performance model for PinT methods
  - Static load balacing
  - Dynamic load balancing

Introduction

- Goals:
  - Performance model for PinT methods
  - Static load balacing
  - Dynamic load balancing
- Challenges:
  - More frameworks than methods (variations with same idea)
  - Large parameter space for each method

Introduction

- Goals:
  - Performance model for PinT methods
  - Static load balacing
  - Dynamic load balancing
- Challenges:
  - More frameworks than methods (variations with same idea)
  - Large parameter space for each method
- Approach:
  - Task-graph based on data-driven formulation of algorithms
  - Allows to cover any implementation using schedules of the tasks
  - Well known tool for load balancing techniques
  - Theoretical lower runtime bound based entirely on the algorithm

Parareal

- Decompose $[0, T]$ into N non-overlapping intervals

$$0 = T_0 < T_1 < ... < T_N = T$$

- Two propagation operators:
  - $\mathcal{F}$ expensive and high accurate
  - $\mathcal{G}$ cheap but less accurate
- Initalization:

$$\mathbf{u}_n^0 = \mathcal{G}(T_n, T_{n-1}, \mathbf{u}_{n-1}^0) \text{ for } n = 1, ..., N,$$

- Parareal iteration:

$$\mathbf{u}_n^{k+1} = \mathcal{F}(T_n, T_{n-1}, \mathbf{u}_{n-1}^k) + \mathcal{G}(T_n, T_{n-1}, \mathbf{u}_{n-1}^{k+1}) - \mathcal{G}(T_n, T_{n-1}, \mathbf{u}_{n-1}^k)$$

for $n = 1, ..., N$ and $k = 0, ...$

Parareal - data driven formulation

Algorithm: Parareal

$\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
for $i \leftarrow 1$ **to** N ▷ Compute initial guess
  $\tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
  $\mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
for $k \leftarrow 1$ **to** N ▷ Parareal iterations
  foreach $i \in \{k, k+1, \ldots, N\}$ **do**
    $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$
  for $i \leftarrow k$ **to** N
    $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$
    $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$

- Typically only $K \ll N$ iterations required

PFASST

- Parallel Full Approximation Scheme in Space and Time (PFASST)

PFASST

- Parallel Full Approximation Scheme in Space and Time (PFASST)
- Sort of Parareal approach, but ...
  - Deferred correction approach instead of fine and coarse propagator
  - Arbitrary hierarchy of levels
  - Coarse level problems are modified using a space-time FAS correction

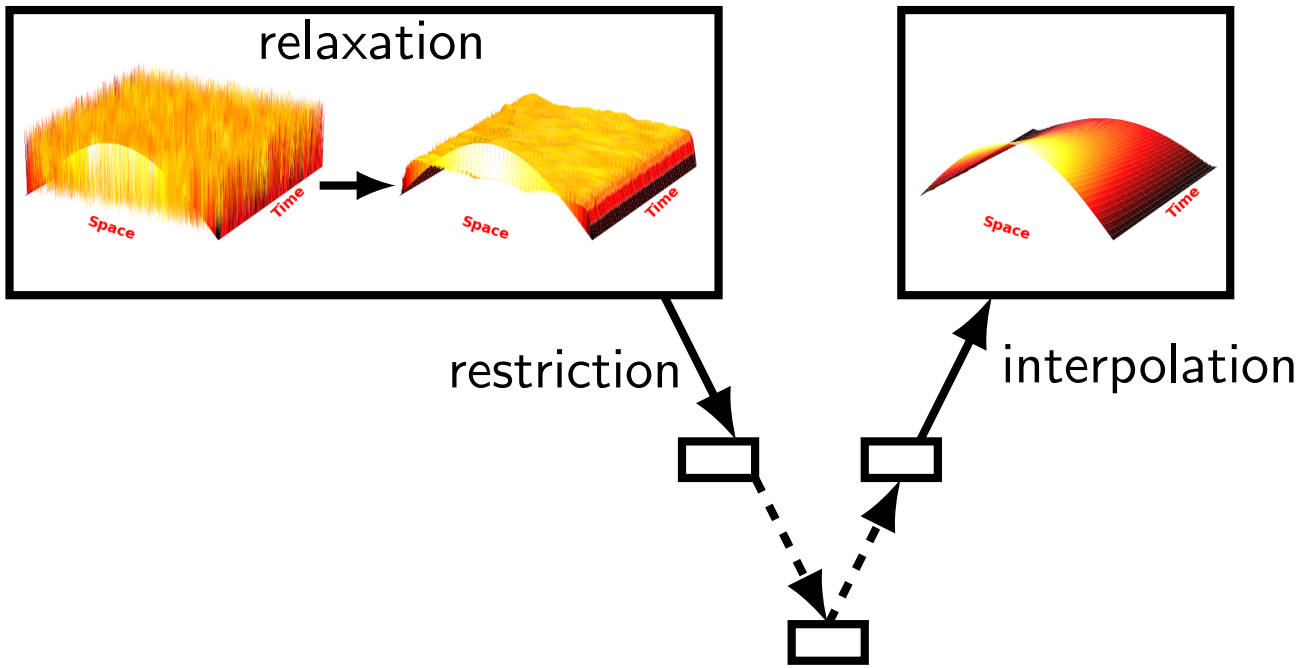
PFASST

- Parallel Full Approximation Scheme in Space and Time (PFASST)
- Sort of Parareal approach, but ...
  - Deferred correction approach instead of fine and coarse propagator
  - Arbitrary hierarchy of levels
  - Coarse level problems are modified using a space-time FAS correction
- Multigrid method
  - Type of space-time multigrid method
  - "Smoother" in the time direction is a spectral deferred correction (SDC) sweep

PFASST

- Parallel Full Approximation Scheme in Space and Time (PFASST)
- Sort of Parareal approach, but ...
  - Deferred correction approach instead of fine and coarse propagator
  - Arbitrary hierarchy of levels
  - Coarse level problems are modified using a space-time FAS correction
- Multigrid method
  - Type of space-time multigrid method
  - "Smoother" in the time direction is a spectral deferred correction (SDC) sweep
- Spectral deferred correction methods
  - Pipelined version of multi-level SDC with each time slice performing SDC sweeps in parallel

Multigrid-reduction-in-time (MGRIT)



- Reduction-based time-multigrid method
- In specialized two-level setting: MGRIT $\equiv$ Parareal

Stopping criterion

- Stopping criterion required for all three iterative methods

Stopping criterion

- Stopping criterion required for all three iterative methods
- Measures the quality of the solution

Stopping criterion

- Stopping criterion required for all three iterative methods
- Measures the quality of the solution
- Various criteria exist:
  - Jump of the approximation between two iterations
  - Finest-level information
  - Residual at single time points
  - Space-time residual
  - ...

Stopping criterion

- Stopping criterion required for all three iterative methods
- Measures the quality of the solution
- Various criteria exist:
  - Jump of the approximation between two iterations
  - Finest-level information
  - Residual at single time points
  - Space-time residual
  - ...
- Criteria hard to compare

Stopping criterion

- Stopping criterion required for all three iterative methods
- Measures the quality of the solution
- Various criteria exist:
  - Jump of the approximation between two iterations
  - Finest-level information
  - Residual at single time points
  - Space-time residual
  - ...
- Criteria hard to compare
- High level categorization into two groups:
  - Local: Measurement of solution quality at individual points in time
  - Global: Global measurement of solution quality

Stopping criterion

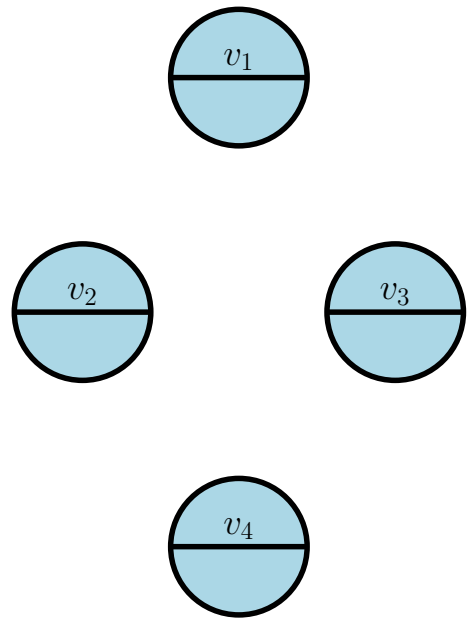
- Stopping criterion required for all three iterative methods
- Measures the quality of the solution
- Various criteria exist:
  - Jump of the approximation between two iterations
  - Finest-level information
  - Residual at single time points
  - Space-time residual
  - ...
- Criteria hard to compare
- High level categorization into two groups:
  - Local: Measurement of solution quality at individual points in time
  - Global: Global measurement of solution quality
- Both can be formulated in a data-driven formulation

Task Graph with communication costs

- Directed acyclic graph $G = (V, E, \omega, c)$

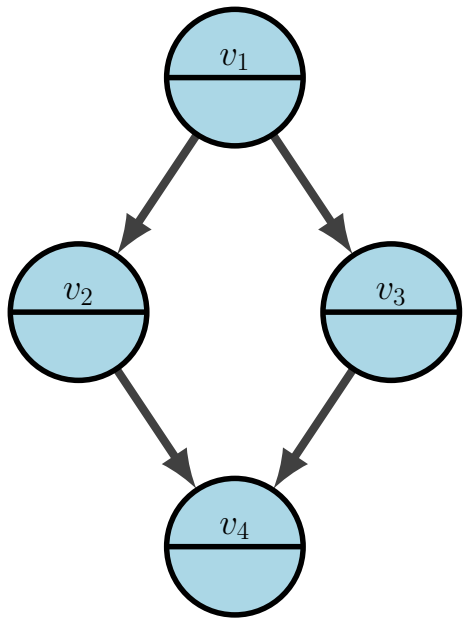
Task Graph with communication costs

- Directed acyclic graph $G = (V, E, \omega, c)$
- Vertices $V = \{v_1, ...v_n\}$
  - Represent tasks



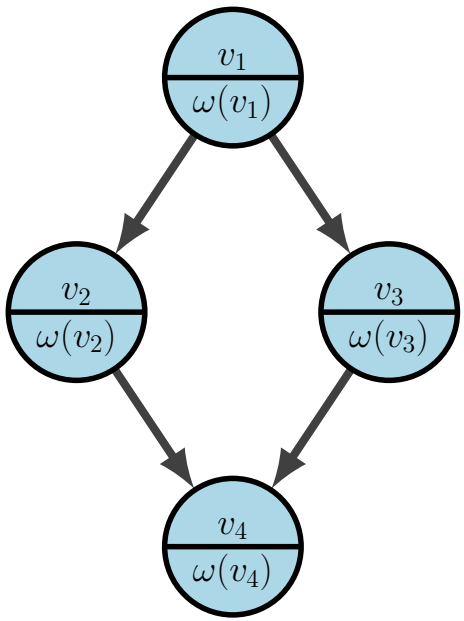
Task Graph with communication costs

- Directed acyclic graph $G = (V, E, \omega, c)$
- Vertices $V = \{v_1, ...v_n\}$
  - Represent tasks
- Directed edges $E \subseteq V \times V$
  - Represent dependencies of tasks



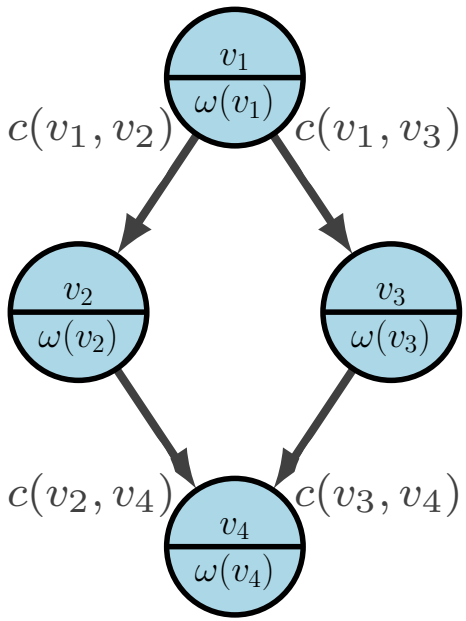
Task Graph with communication costs

- Directed acyclic graph $G = (V, E, \omega, c)$
- Vertices $V = \{v_1, ...v_n\}$
  - Represent tasks
- Directed edges $E \subseteq V \times V$
  - Represent dependencies of tasks
- Node weights $\omega : V \to \mathbb{R}_0^+$
  - Represent computational cost of tasks



Task Graph with communication costs

- Directed acyclic graph $G = (V, E, \omega, c)$
- Vertices $V = \{v_1, ...v_n\}$
  - Represent tasks
- Directed edges $E \subseteq V \times V$
  - Represent dependencies of tasks
- Node weights $\omega : V \to \mathbb{R}_0^+$
  - Represent computational cost of tasks
- Edge weights $c : E \to \mathbb{R}_0^+$
  - Represent communication cost between tasks



Allocation and schedule

- $P = \{p_1, \ldots, p_{N_P}\}$
  - Set of N_P processes

Allocation and schedule

- $P = \{p_1, \ldots, p_{N_P}\}$
  - Set of N_P processes
- Allocation function $A : V \to P$
  - Assigns each task in V to a process

Allocation and schedule

- $P = \{p_1, \ldots, p_{N_P}\}$
  - Set of N_P processes
- Allocation function $A : V \to P$
  - Assigns each task in V to a process
- Schedule $S : V \to \mathbb{R}_0^+$
  - Assigns a starting point to each task, subject to the constraints:
    - $\forall (v_i, v_j) \in E, S(v_j) \geq S(v_i) + \omega(v_i) + c(v_i, v_j)$
    - $\forall v_i, v_j \in V, v_i \neq v_j, A(v_i) = A(v_j) \Rightarrow S(v_i) \geq S(v_j) + \omega(v_j) \vee S(v_j) \geq S(v_i) + \omega(v_i)$

Allocation and schedule

- $P = \{p_1, \ldots, p_{N_P}\}$
  - Set of N_P processes
- Allocation function $A : V \to P$
  - Assigns each task in V to a process
- Schedule $S : V \to \mathbb{R}_0^+$
  - Assigns a starting point to each task, subject to the constraints:
    - $\forall (v_i, v_j) \in E, S(v_j) \geq S(v_i) + \omega(v_i) + c(v_i, v_j)$
    - $\forall v_i, v_j \in V, v_i \neq v_j, A(v_i) = A(v_j) \Rightarrow S(v_i) \geq S(v_j) + \omega(v_j) \vee S(v_j) \geq S(v_i) + \omega(v_i)$
- Makespan or runtime of a given allocation and schedule:

$$\max_{v \in V}(S(v) + \omega(v))$$

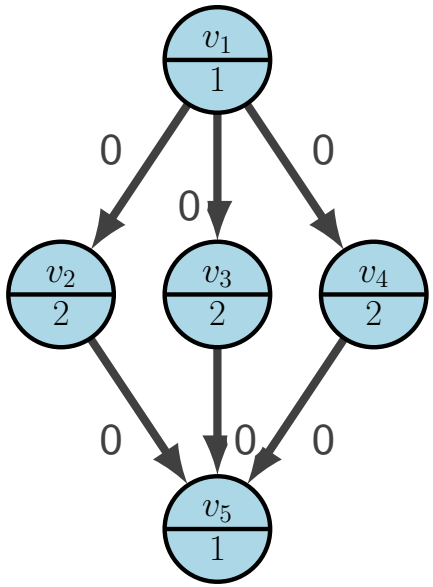
Allocation and schedule

- $P = \{p_1, \ldots, p_{N_P}\}$
  - Set of N_P processes
- Allocation function $A : V \to P$
  - Assigns each task in V to a process
- Schedule $S : V \to \mathbb{R}_0^+$
  - Assigns a starting point to each task, subject to the constraints:
    - $\forall (v_i, v_j) \in E, S(v_j) \geq S(v_i) + \omega(v_i) + c(v_i, v_j)$
    - $\forall v_i, v_j \in V, v_i \neq v_j, A(v_i) = A(v_j) \Rightarrow S(v_i) \geq S(v_j) + \omega(v_j) \vee S(v_j) \geq S(v_i) + \omega(v_i)$
- Makespan or runtime of a given allocation and schedule:

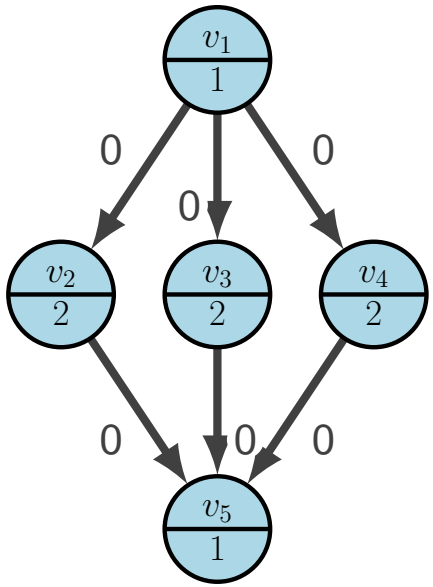
$$\max_{v \in V}(S(v) + \omega(v))$$

- If $c(e) = 0 \; \forall e \in E$: Minimum possible makespan for $N_P = \infty$ is longest path within the graph

Schedule example



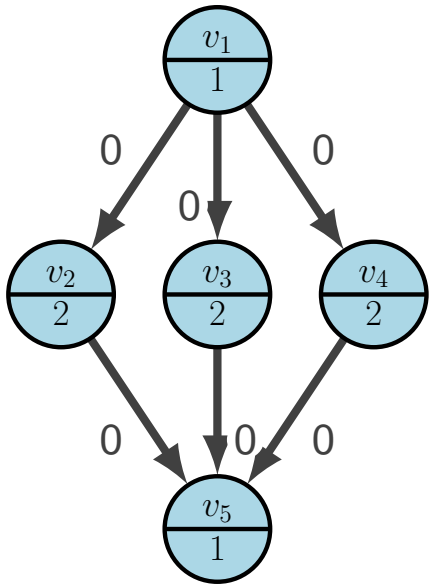
Schedule example



- $N_P = 2$

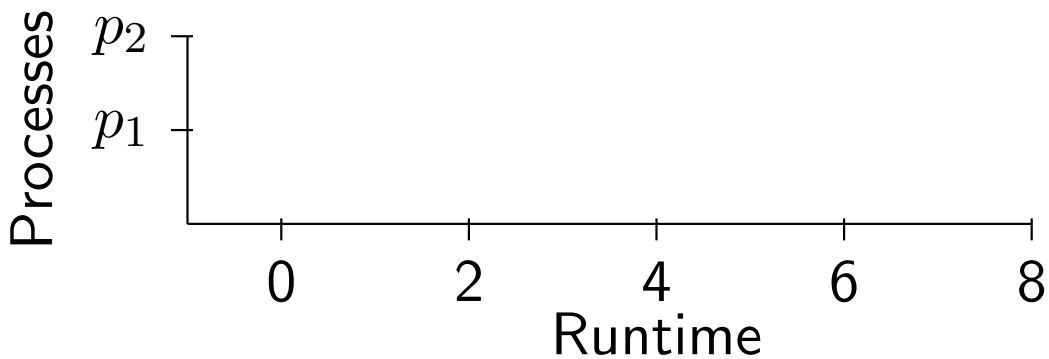
A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

Schedule example

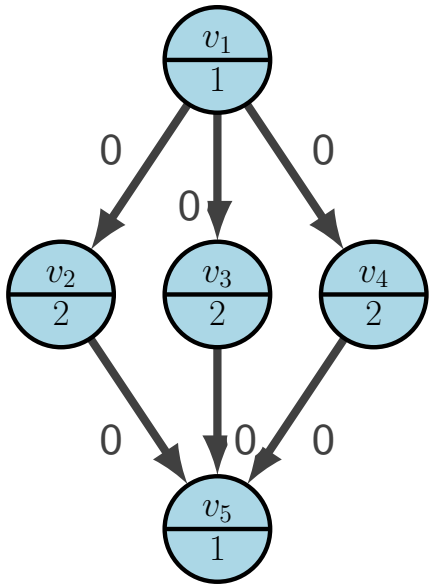


- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

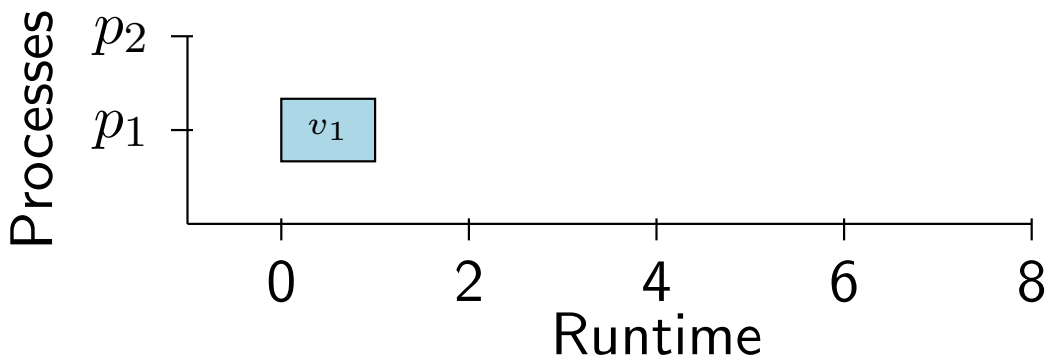


Schedule example

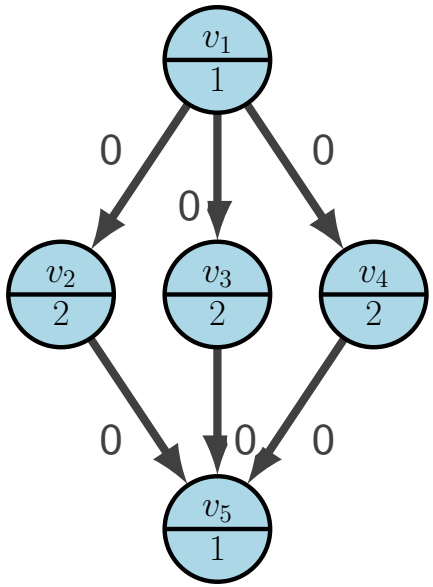


- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

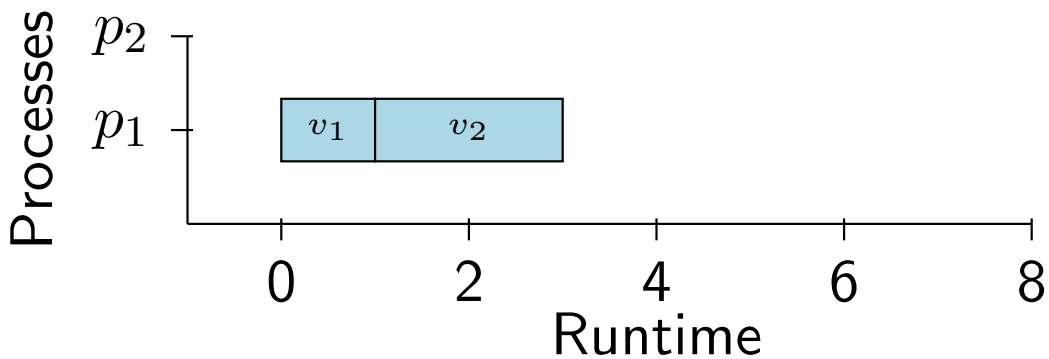


Schedule example

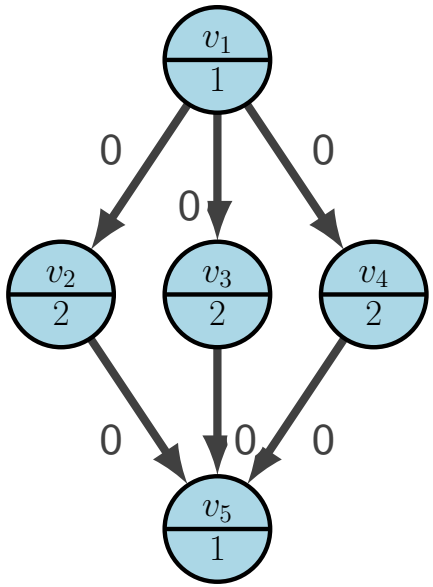


- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

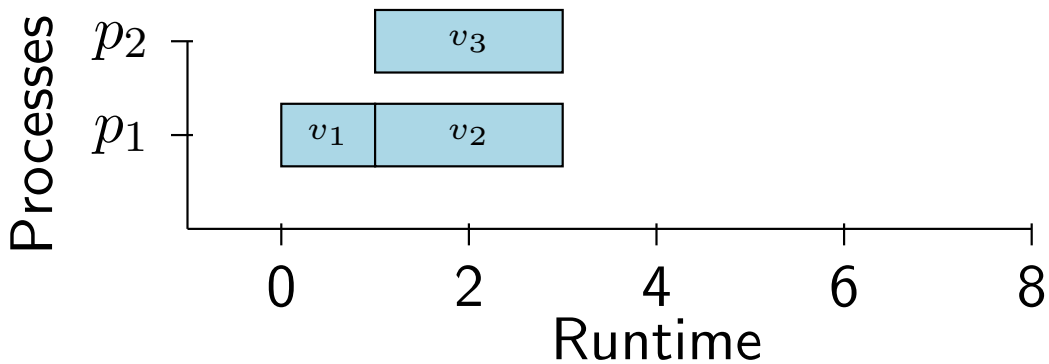


Schedule example

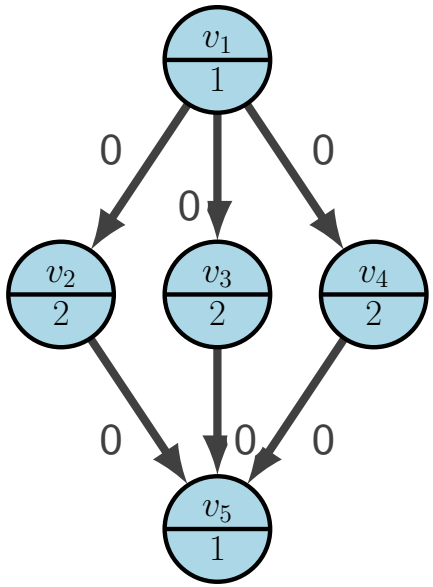


- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

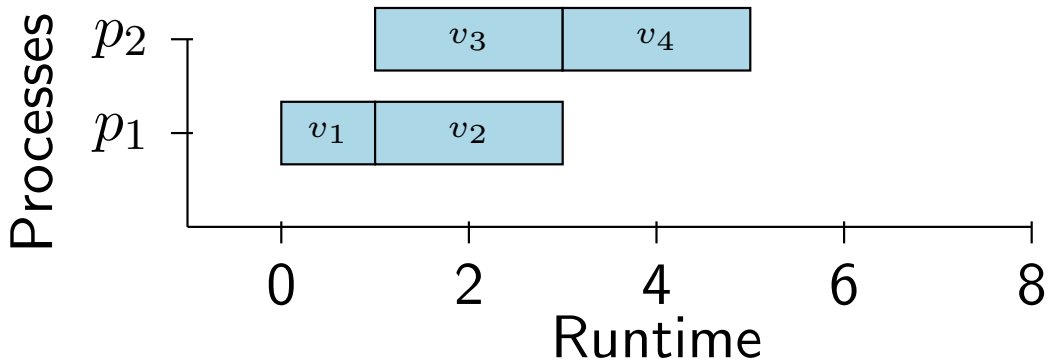


Schedule example

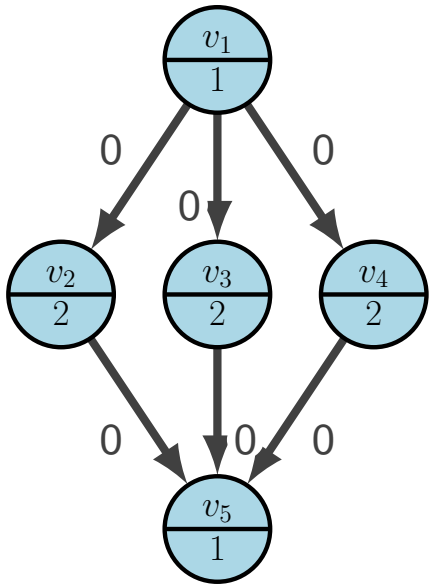


- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

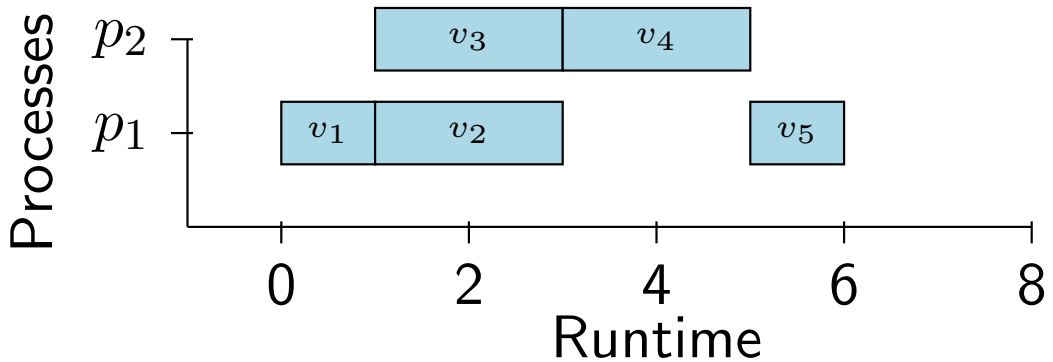


Schedule example

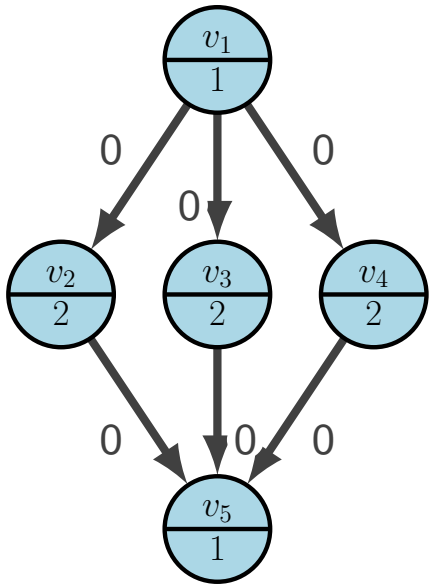


- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$

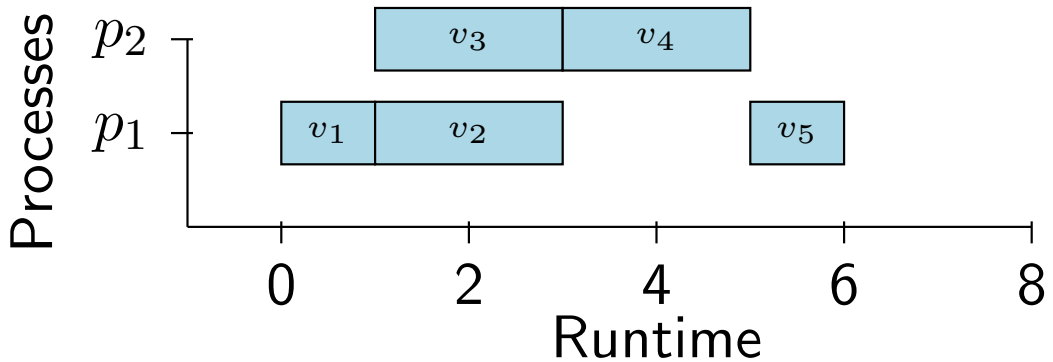


Schedule example



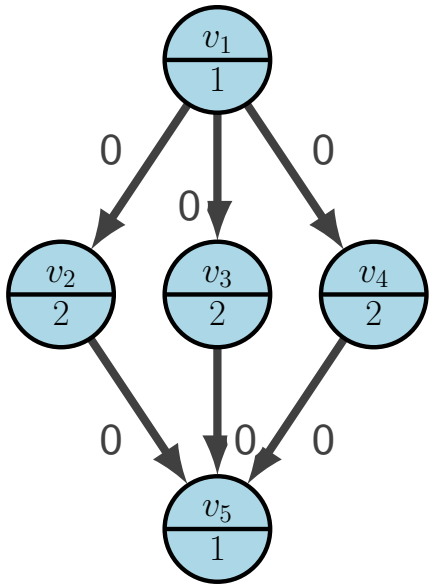
- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$



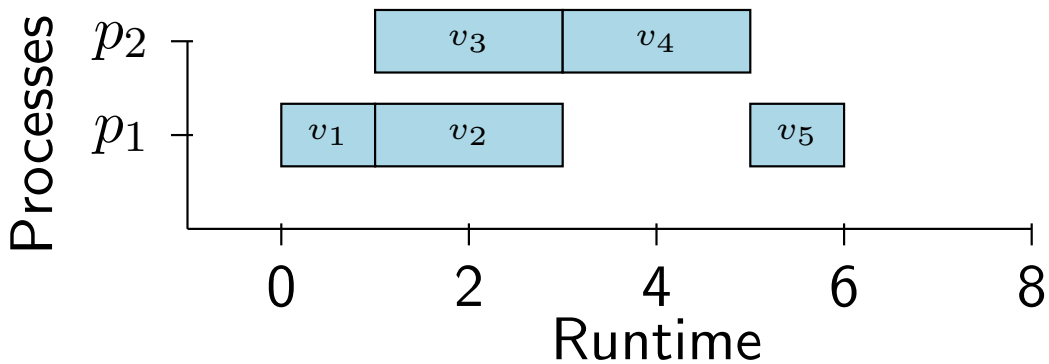
- Makespan of schedule: 6

Schedule example



- $N_P = 2$

A(v)
$v_1 \rightarrow p_1$
$v_2 \rightarrow p_1$
$v_3 \rightarrow p_2$
$v_4 \rightarrow p_2$
$v_5 \rightarrow p_1$



- Makespan of schedule: 6
- Longest path $\{v_1 \rightarrow v_2 \rightarrow v_5\} \Rightarrow$ minimal parallel runtime: 4

Task graph for Parareal $N = 3$

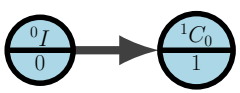
Algorithm: Parareal

1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ Initial guess
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...

Task graph for Parareal $N = 3$

Algorithm: Parareal

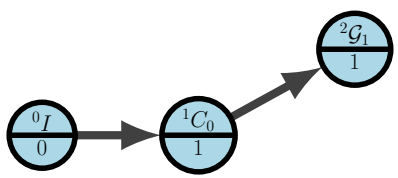
1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ Initial guess
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

Algorithm: Parareal

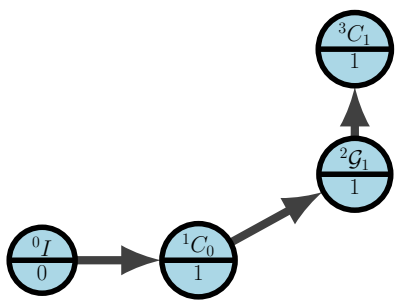
1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ $i = 1$
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

Algorithm: Parareal

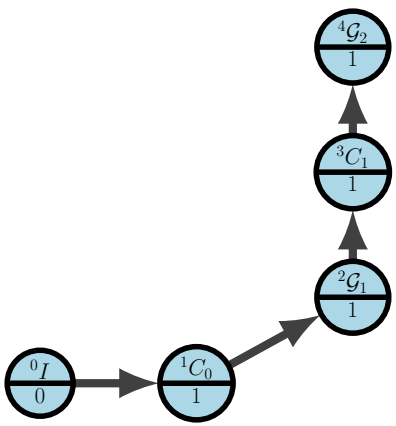
1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ $i = 1$
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

Algorithm: Parareal

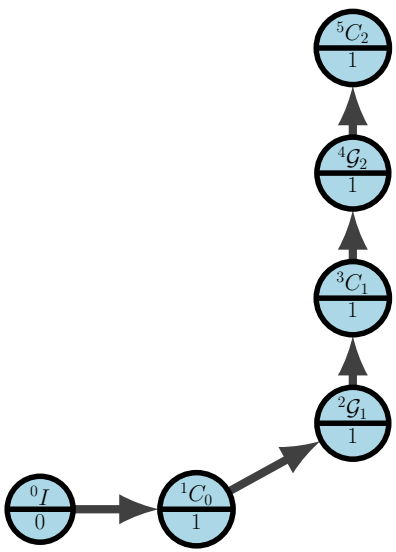
1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ $i = 2$
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

Algorithm: Parareal

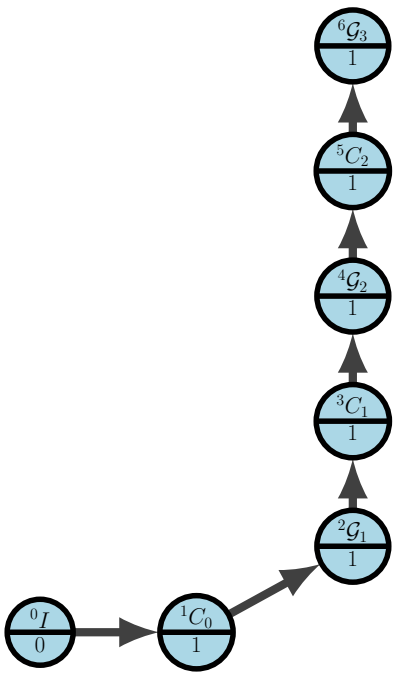
1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ $i = 2$
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

Algorithm: Parareal

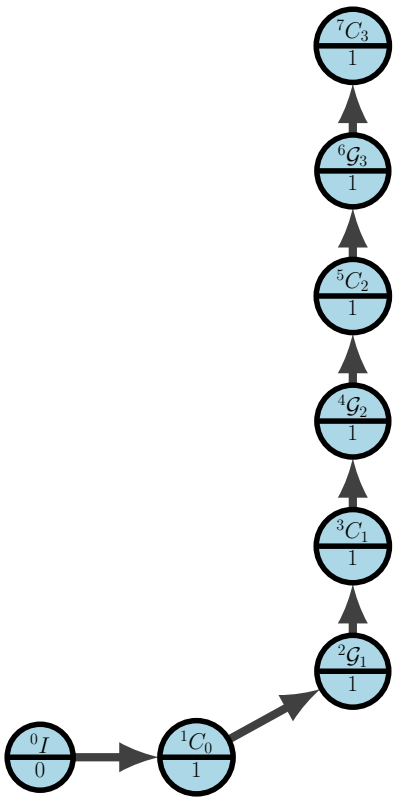
1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ $i = 3$
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

Algorithm: Parareal

1 $\mathbf{u}_0^0 \leftarrow \mathbf{u}_0$
2 **for** $i \leftarrow 1$ **to** N ▷ $i = 3$
3 $\quad \tilde{\mathbf{u}}_i^0 \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^0)$
4 $\quad \mathbf{u}_i^0 \leftarrow \tilde{\mathbf{u}}_i^0$
5 ...



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

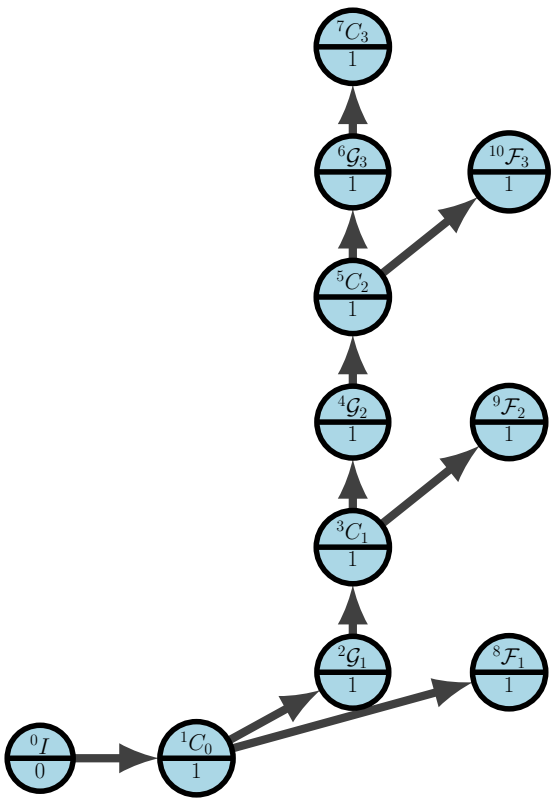
6 | **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 | | $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 | **for** $i \leftarrow k$ **to** N

9 | | $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 | | $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

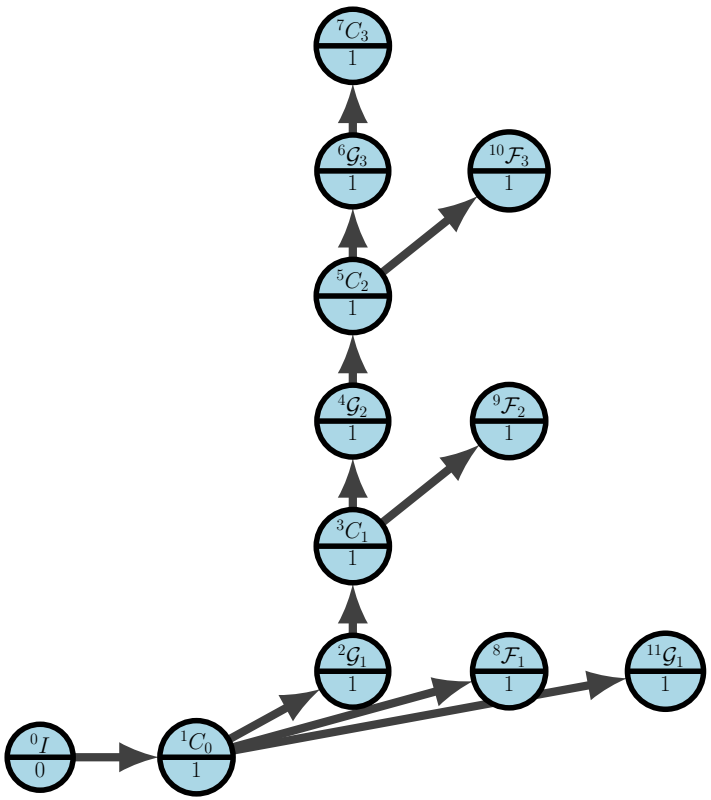
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 1$

9 $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

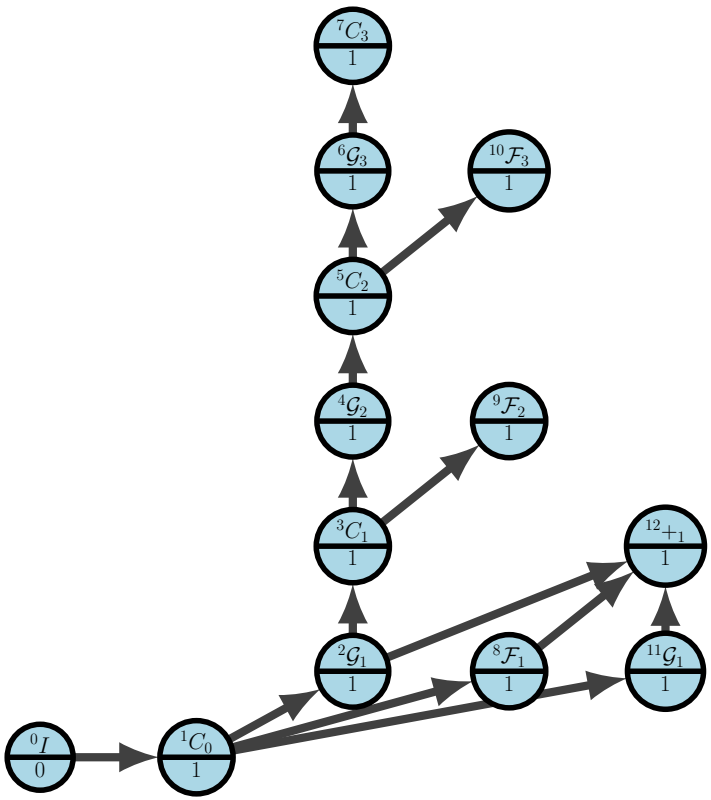
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 1$

9 $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

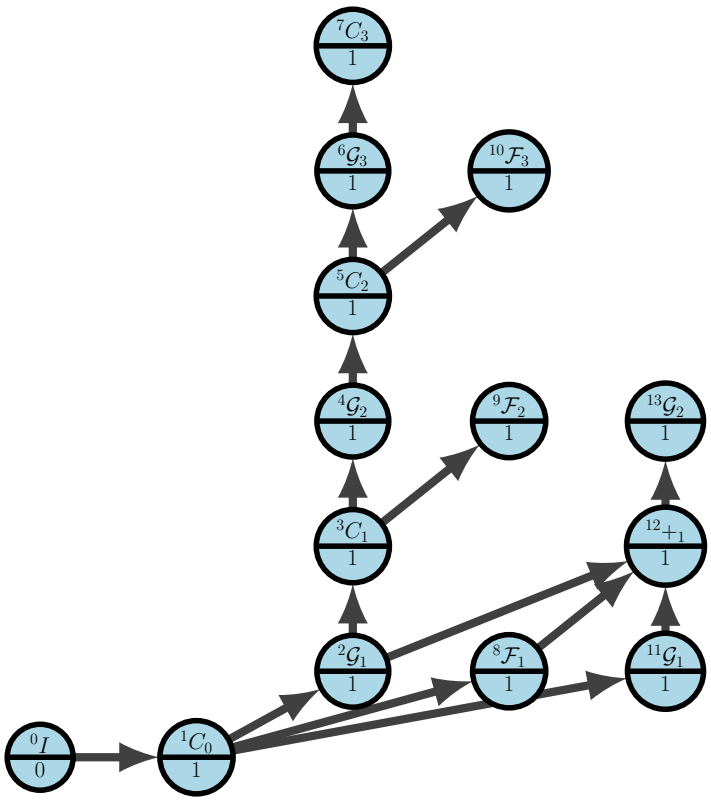
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 2$

9 $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

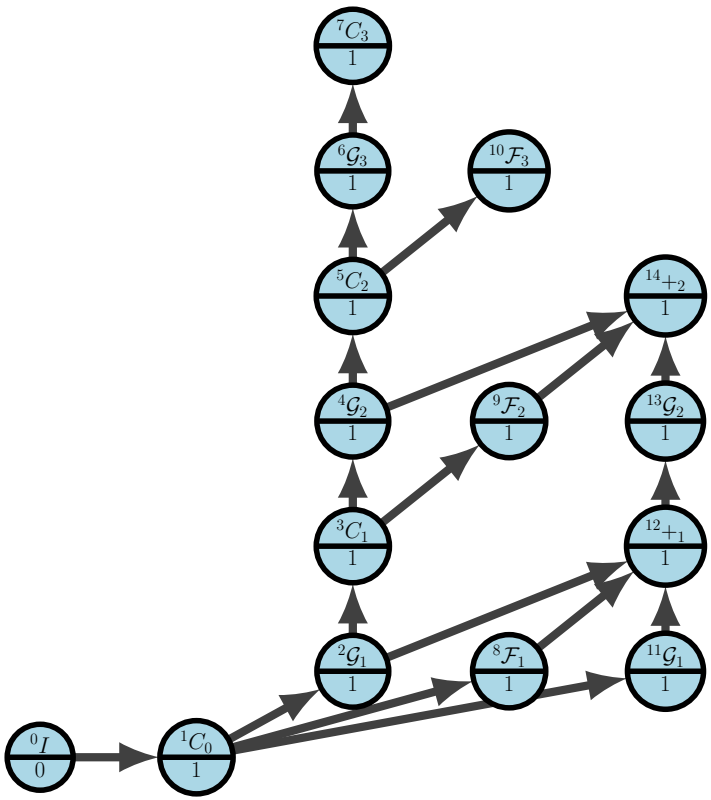
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 2$

9 $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

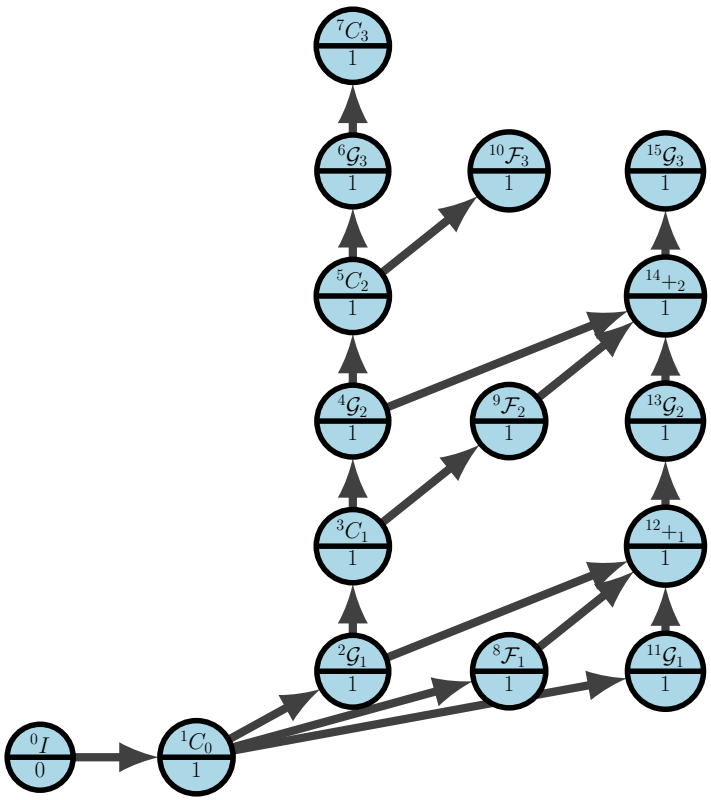
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 3$

9 $\tilde{\mathbf{u}}_i^k \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^k \leftarrow \tilde{\mathbf{u}}_i^k + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 1$

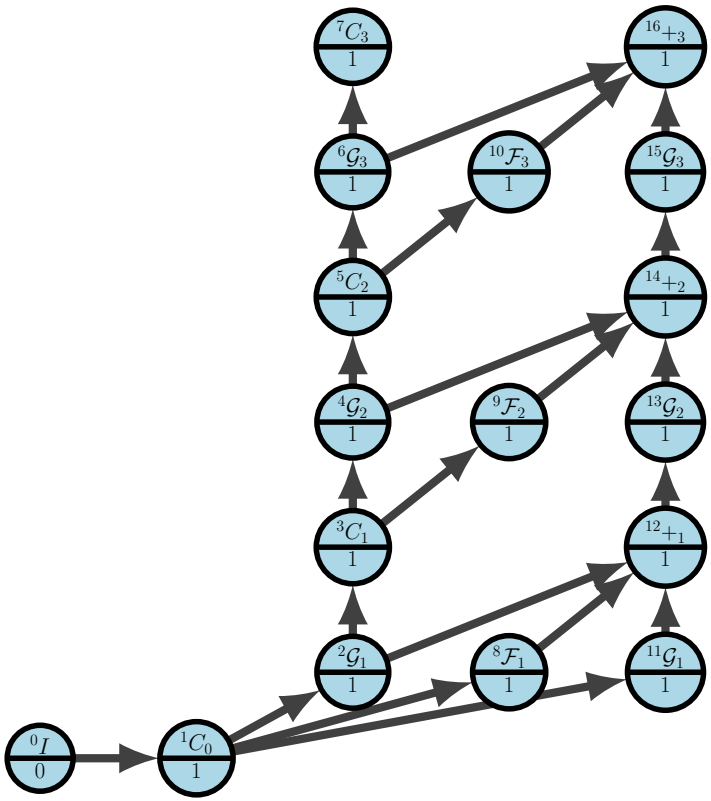
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 3$

9 $\tilde{\mathbf{u}}_i^{k} \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^{k} \leftarrow \tilde{\mathbf{u}}_i^{k} + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



Task graph for Parareal $N = 3$

5 **for** $k \leftarrow 1$ **to** N ▷ $k = 3$

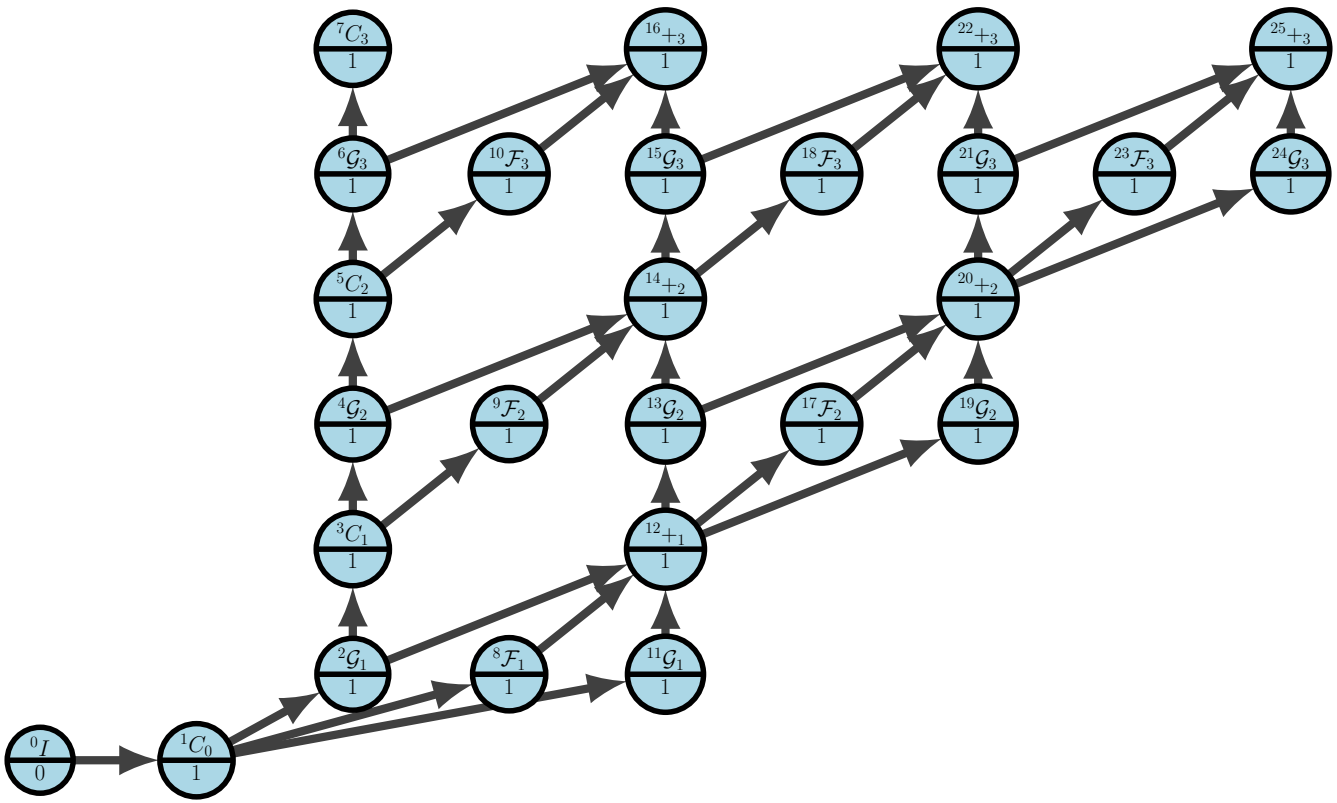
6 **foreach** $i \in \{k, k+1, \ldots, N\}$ **do**

7 $\hat{\mathbf{u}}_i^{k-1} \leftarrow \mathcal{F}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{k-1})$

8 **for** $i \leftarrow k$ **to** N ▷ $i = 3$

9 $\tilde{\mathbf{u}}_i^{k} \leftarrow \mathcal{G}(T_i, T_{i-1}, \mathbf{u}_{i-1}^{\min(k,i-1)})$

10 $\mathbf{u}_i^{k} \leftarrow \tilde{\mathbf{u}}_i^{k} + \hat{\mathbf{u}}_i^{k-1} - \tilde{\mathbf{u}}_i^{k-1}$



PinT parallelization strategies

- Typically static allocation of time points to processes

PinT parallelization strategies

- Typically static allocation of time points to processes
- Scheduling based on this allocation

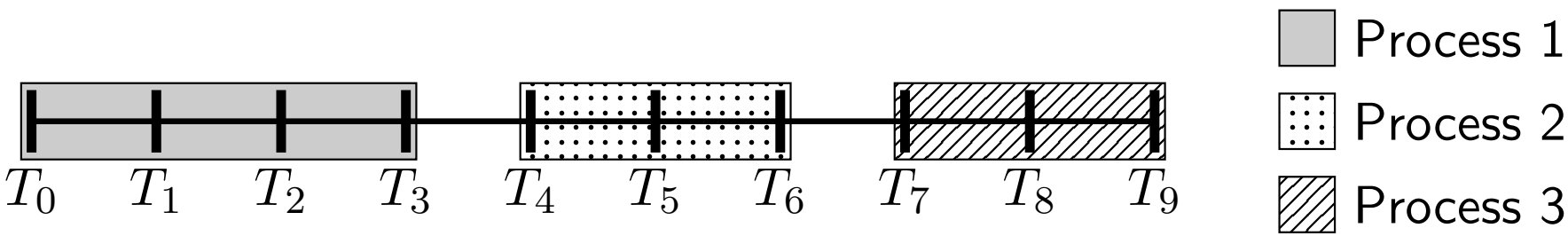
PinT parallelization strategies

- Typically static allocation of time points to processes
- Scheduling based on this allocation
- Two types of allocation:

PinT parallelization strategies

- Typically static allocation of time points to processes
- Scheduling based on this allocation
- Two types of allocation:
  - Block-by-block basis (each block roughly same number of time points)

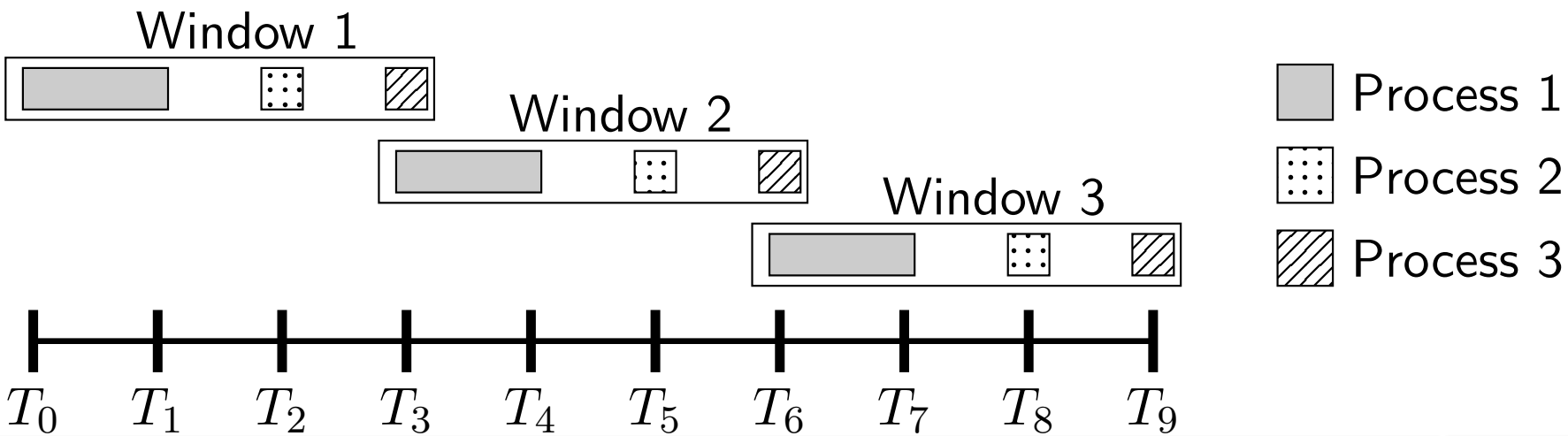
- Example: $N = 9$, $N_P = 3$



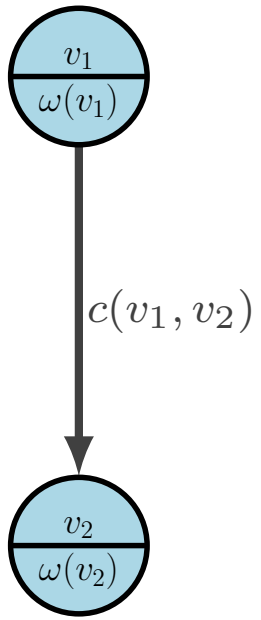
PinT parallelization strategies

- Typically static allocation of time points to processes
- Scheduling based on this allocation
- Two types of allocation:
  - Block-by-block basis (each block roughly same number of time points)
  - Windowing with block-by-block (applying method multiple times)

- Example: $N = 9$, $N_P = 3$

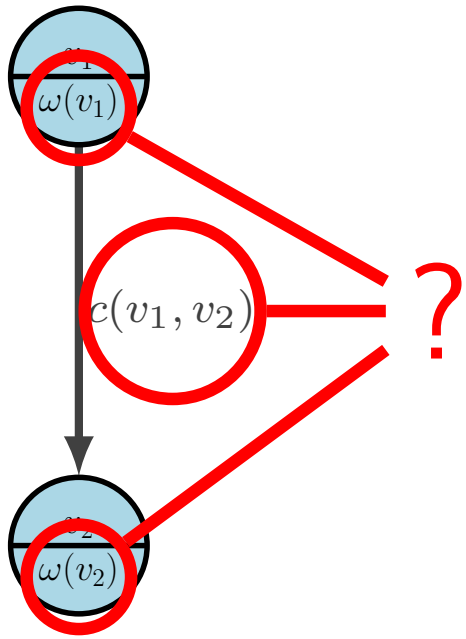


Weighting of tasks and communication



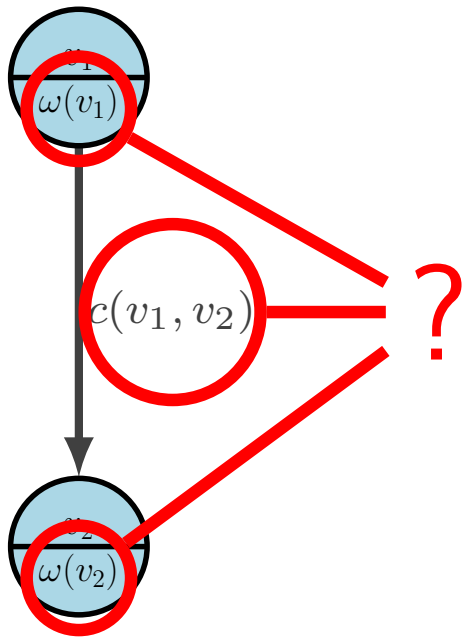
Weighting of tasks and communication

- How to set the costs?



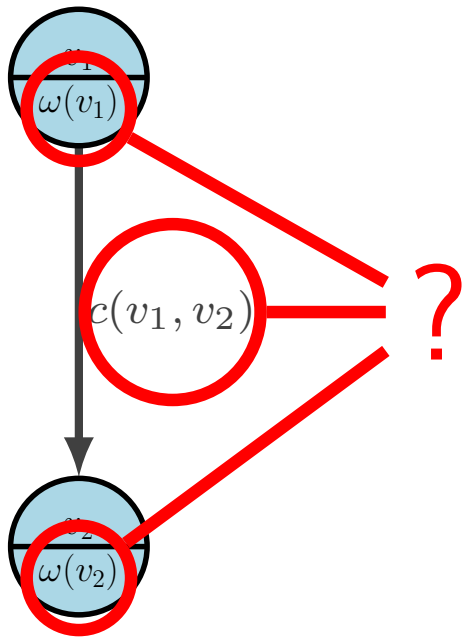
Weighting of tasks and communication

- How to set the costs?
- Two ways:
  - Theoretical model:
    - \+ Exact and theoretical analysis
    - \+ Independent of implementation
    - – Huge effort
    - – Not very flexible, models one problem
  - Measure runtimes of operations:
    - Measure any type of operation
    - \+ Easy and flexible
    - – Requires implementation



Weighting of tasks and communication

- How to set the costs?
- Two ways:
  - Theoretical model:
    - \+ Exact and theoretical analysis
    - \+ Independent of implementation
    - – Huge effort
    - – Not very flexible, models one problem
  - Measure runtimes of operations:
    - Measure any type of operation
    - \+ Easy and flexible
    - – Requires implementation
- Decision depends on use case



Comparison model and PinT libraries

- Compare model with four Pint libraries:
  - LibPFASST[1]: Fortran implementation of PFASST
  - PySDC[2]: Python implementation of PFASST
  - XBRAID[3]: C implementation of MGRIT
  - PyMGRIT[4]: Python implementation of MGRIT

[1] `https://github.com/libpfasst/LibPFASST`
[2] `https://github.com/Parallel-in-Time/pySDC`
[3] `https://github.com/XBraid/xbraid`
[4] `https://github.com/pymgrit/pymgrit`

Comparison model and PinT libraries

- Compare model with four Pint libraries:
  - LibPFASST[1]: Fortran implementation of PFASST
  - PySDC[2]: Python implementation of PFASST
  - XBRAID[3]: C implementation of MGRIT
  - PyMGRIT[4]: Python implementation of MGRIT
- Each library has a large set of pre-implemented features

[1] `https://github.com/libpfasst/LibPFASST`
[2] `https://github.com/Parallel-in-Time/pySDC`
[3] `https://github.com/XBraid/xbraid`
[4] `https://github.com/pymgrit/pymgrit`

Comparison model and PinT libraries

- Compare model with four Pint libraries:
  - LibPFASST[1]: Fortran implementation of PFASST
  - PySDC[2]: Python implementation of PFASST
  - XBRAID[3]: C implementation of MGRIT
  - PyMGRIT[4]: Python implementation of MGRIT
- Each library has a large set of pre-implemented features
- Each library requires problem-dependent functions

[1] `https://github.com/libpfasst/LibPFASST`
[2] `https://github.com/Parallel-in-Time/pySDC`
[3] `https://github.com/XBraid/xbraid`
[4] `https://github.com/pymgrit/pymgrit`

Comparison model and PinT libraries

- Compare model with four Pint libraries:
  - LibPFASST[1]: Fortran implementation of PFASST
  - PySDC[2]: Python implementation of PFASST
  - XBRAID[3]: C implementation of MGRIT
  - PyMGRIT[4]: Python implementation of MGRIT
- Each library has a large set of pre-implemented features
- Each library requires problem-dependent functions
- Challenging to choose problem
  - Different languages
  - Methods with different strengths

[1] `https://github.com/libpfasst/LibPFASST`
[2] `https://github.com/Parallel-in-Time/pySDC`
[3] `https://github.com/XBraid/xbraid`
[4] `https://github.com/pymgrit/pymgrit`

Comparison model and PinT libraries

- Compare model with four Pint libraries:
  - LibPFASST[1]: Fortran implementation of PFASST
  - PySDC[2]: Python implementation of PFASST
  - XBRAID[3]: C implementation of MGRIT
  - PyMGRIT[4]: Python implementation of MGRIT
- Each library has a large set of pre-implemented features
- Each library requires problem-dependent functions
- Challenging to choose problem
  - Different languages
  - Methods with different strengths
- Pseudo spatial problem:
  - Each required function sleeps only for a controllable time
  - Enables large test sets in terms of task costs

[1] `https://github.com/libpfasst/LibPFASST`
[2] `https://github.com/Parallel-in-Time/pySDC`
[3] `https://github.com/XBraid/xbraid`
[4] `https://github.com/pymgrit/pymgrit`

PFASST - setting

- LibPFASST
  - Classical view
  - Burn-in prediction phase
- PySDC
  - Multigrid view
  - Fine-sweep prediction phase
- Both:
  - Local convergence criterion
  - Implicit sweeper (pre-implemented)
  - Cost per implicit solve: 0.05 s
  - All other costs: 0 s
  - Runtime of operations is measured (reusable for other problems)
  - Communication cost: 0

Parameters	
Number of time intervals	N
Number processes time dimension	N_P
Number of levels	L
Number of iterations	K
Convergence criterion	
Two "views"	
Number of sweeps on level ℓ	
Collocation nodes on level ℓ	
Predictor type	
Skip fine-level sweep at start/end	
Runtime communication in time	
Runtime convergence criterion	
Runtime `SDCSweep` at level ℓ	
Runtime `FEvalAll` at level ℓ	
Runtime `FEvalSingle` at level ℓ	
Runtime `RestrictAll` at level ℓ	
Runtime `RestrictSingle` at level ℓ	
Runtime `InterpolateAll` at level ℓ	
Runtime `InterpolateSingle` at level ℓ	
Runtime `FAS` at level ℓ	

LibPFASST - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	`LibPFASST` runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.06	5.05	5.04
16	16	2	10	$(1, 1)$	$(7, 5)$	8.88	8.87	8.86
16	16	2	10	$(2, 1)$	$(5, 3)$	7.07	7.04	7.03
16	16	2	10	$(2, 1)$	$(7, 5)$	11.9	11.89	11.88
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.16	12.12	12.11
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	15.17	15.13	15.12
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	16.17	16.14	16.13
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.17	14.14	14.13
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	18.18	18.14	18.13

LibPFASST - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	`LibPFASST` runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.06	5.05	5.04
16	16	2	10	$(1, 1)$	$(7, 5)$	8.88	8.87	8.86
16	16	2	10	$(2, 1)$	$(5, 3)$	7.07	7.04	7.03
16	16	2	10	$(2, 1)$	$(7, 5)$	11.9	11.89	11.88
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.16	12.12	12.11
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	15.17	15.13	15.12
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	16.17	16.14	16.13
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.17	14.14	14.13
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	18.18	18.14	18.13

- Actual runtime very close to predictions

LibPFASST - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	`LibPFASST` runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.06	5.05	5.04
16	16	2	10	$(1, 1)$	$(7, 5)$	8.88	8.87	8.86
16	16	2	10	$(2, 1)$	$(5, 3)$	7.07	7.04	7.03
16	16	2	10	$(2, 1)$	$(7, 5)$	11.9	11.89	11.88
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.16	12.12	12.11
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	15.17	15.13	15.12
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	16.17	16.14	16.13
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.17	14.14	14.13
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	18.18	18.14	18.13

- Actual runtime very close to predictions
- Lower bound (L.b.) and predictions very similar

PySDC - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	pySDC runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.22	5.00	5.00
16	16	2	10	$(1, 1)$	$(7, 5)$	9.45	8.99	8.99
16	16	2	10	$(2, 1)$	$(5, 3)$	7.38	7.07	7.07
16	16	2	10	$(2, 1)$	$(7, 5)$	12.78	12.23	12.23
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.82	12.34	12.34
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	16.23	15.57	15.56
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	17.16	16.41	16.30
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.96	14.26	13.22
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	19.38	18.57	15.43

PySDC - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	pySDC runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.22	5.00	5.00
16	16	2	10	$(1, 1)$	$(7, 5)$	9.45	8.99	8.99
16	16	2	10	$(2, 1)$	$(5, 3)$	7.38	7.07	7.07
16	16	2	10	$(2, 1)$	$(7, 5)$	12.78	12.23	12.23
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.82	12.34	12.34
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	16.23	15.57	15.56
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	17.16	16.41	16.30
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.96	14.26	13.22
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	19.38	18.57	15.43

- Actual runtime differs by no more than 5%

PySDC - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	pySDC runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.22	5.00	5.00
16	16	2	10	$(1, 1)$	$(7, 5)$	9.45	8.99	8.99
16	16	2	10	$(2, 1)$	$(5, 3)$	7.38	7.07	7.07
16	16	2	10	$(2, 1)$	$(7, 5)$	12.78	12.23	12.23
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.82	12.34	12.34
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	16.23	15.57	15.56
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	17.16	16.41	16.30
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.96	14.26	13.22
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	19.38	18.57	15.43

- Actual runtime differs by no more than 5%
- Lower bound and predictions similiar for most settings

PySDC - model vs. runtime

N	N_P	L	K	# Sweeps	collocation nodes	pySDC runtime	Model	L. b. ($N_P = \infty$)
16	16	2	10	$(1, 1)$	$(5, 3)$	5.22	5.00	5.00
16	16	2	10	$(1, 1)$	$(7, 5)$	9.45	8.99	8.99
16	16	2	10	$(2, 1)$	$(5, 3)$	7.38	7.07	7.07
16	16	2	10	$(2, 1)$	$(7, 5)$	12.78	12.23	12.23
32	32	4	10	$(1, 1, 1, 1)$	$(7, 5, 3, 2)$	12.82	12.34	12.34
32	32	4	10	$(2, 1, 1, 1)$	$(7, 5, 3, 2)$	16.23	15.57	15.56
32	32	4	10	$(1, 2, 1, 1)$	$(7, 5, 3, 2)$	17.16	16.41	16.30
32	32	4	10	$(1, 1, 2, 1)$	$(7, 5, 3, 2)$	14.96	14.26	13.22
32	32	4	10	$(1, 2, 2, 1)$	$(7, 5, 3, 2)$	19.38	18.57	15.43

- Actual runtime differs by no more than 5%
- Lower bound and predictions similiar for most settings
- Small derivations for some four-level settings (PFASST multigrid view)

MGRIT - setting

- XBRAID
  - Global convergence criterion
- PyMGRIT
  - Local and global convergence criteria
- Both:
  - Cost per time integration: 0.05 s
  - All other costs: 0 s
  - Runtime of operations is measured (reusable for other problems)
  - Communication cost: 0

Parameters	
Number of time intervals	N
Number processes time dimension	N_P
Number of levels	L
Number of iterations	K
Convergence criterion	
Coarsening factor from level ℓ to level $\ell + 1$	
Cycle type	
Nested iterations	
Skip down	
Number of CF-relaxations on level ℓ	
Runtime communication in time	
Runtime convergence criterion	
Runtime time integrator at level ℓ	
Runtime spatial restriction at level ℓ	
Runtime spatial interpolation at level ℓ	

XBRAID - model vs. runtime

N	N_P	L	K	cyc.	coarsening	$\#\ CF$-relax.	skip. down	XBRAID runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	True	19.02	18.93	15.61
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	False	22.79	21.83	17.91
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	True	21.59	21.51	13.63
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	False	23.90	23.01	14.99
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	True	22.81	22.66	11.88
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	False	26.38	26.16	13.43
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	True	30.71	30.64	15.17
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	False	32.61	32.49	16.04

XBRAID - model vs. runtime

N	N_P	L	K	cyc.	coarsening	# CF-relax.	skip. down	XBRAID runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	True	19.02	18.93	15.61
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	False	22.79	21.83	17.91
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	True	21.59	21.51	13.63
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	False	23.90	23.01	14.99
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	True	22.81	22.66	11.88
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	False	26.38	26.16	13.43
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	True	30.71	30.64	15.17
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	False	32.61	32.49	16.04

- Actual runtime very close to predictions

XBRAID - model vs. runtime

N	N_P	L	K	cyc.	coarsening	$\#\ CF$-relax.	skip. down	XBRAID runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	True	19.02	18.93	15.61
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	False	22.79	21.83	17.91
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	True	21.59	21.51	13.63
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	False	23.90	23.01	14.99
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	True	22.81	22.66	11.88
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	False	26.38	26.16	13.43
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	True	30.71	30.64	15.17
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	False	32.61	32.49	16.04

- Actual runtime very close to predictions
  - Cluster based problem where the first communication between nodes is especially expensive

XBRAID - model vs. runtime

N	N_P	L	K	cyc.	coarsening	$\#\ CF$-relax.	skip. down	XBRAID runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	True	19.02	18.93	15.61
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	False	22.79	21.83	17.91
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	True	21.59	21.51	13.63
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	False	23.90	23.01	14.99
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	True	22.81	22.66	11.88
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	False	26.38	26.16	13.43
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	True	30.71	30.64	15.17
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	False	32.61	32.49	16.04

- Actual runtime very close to predictions
  - Cluster based problem where the first communication between nodes is especially expensive
- Theoretically further potential
  - Largely due to global convergence criterion (blocking communication)

XBRAID - model vs. runtime

N	N_P	L	K	cyc.	coarsening	$\#\ CF$-relax.	skip. down	XBRAID runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	True	19.02	18.93	15.61
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	False	22.79	21.83	17.91
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	True	21.59	21.51	13.63
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	False	23.90	23.01	14.99
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	True	22.81	22.66	11.88
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	False	26.38	26.16	13.43
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	True	30.71	30.64	15.17
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	False	32.61	32.49	16.04

- Actual runtime very close to predictions
  - Cluster based problem where the first communication between nodes is especially expensive
- Theoretically further potential
  - Largely due to global convergence criterion (blocking communication)
  - Comparison between 256 and ∞ processes

PyMGRIT - model vs. runtime

N	N_P	L	K	cyc.	coarsening	# CF-relax.	nest. iter.	conv. crit.	PyMGRIT runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	T	local	19.99	19.53	18.01
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	F	local	18.87	18.43	18.02
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	T	local	17.69	17.17	15.09
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	F	local	16.98	16.66	15.09
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	T	global	24.66	23.91	18.06
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	F	global	23.28	21.94	18.01
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	T	global	25.08	24.04	15.07
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	F	global	24.24	23.15	15.08
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	T	local	25.82	25.36	13.52
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	T	local	26.06	25.43	16.13
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	T	global	30.91	29.97	13.49
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	T	global	35.93	34.82	16.14

PyMGRIT - model vs. runtime

N	N_P	L	K	cyc.	coarsening	# CF-relax.	nest. iter.	conv. crit.	PyMGRIT runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	T	local	19.99	19.53	18.01
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	F	local	18.87	18.43	18.02
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	T	local	17.69	17.17	15.09
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	F	local	16.98	16.66	15.09
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	T	global	24.66	23.91	18.06
4096	256	5	6	V	$(16, 4, 4, 4)$	$(1, 1, 1, 1)$	F	global	23.28	21.94	18.01
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	T	global	25.08	24.04	15.07
4096	256	5	6	F	$(16, 4, 4, 4)$	$(0, 0, 0, 0)$	F	global	24.24	23.15	15.08
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	T	local	25.82	25.36	13.52
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	T	local	26.06	25.43	16.13
4096	256	6	6	V	$(4, 4, 4, 4, 4)$	$(1, 1, 1, 1, 1)$	T	global	30.91	29.97	13.49
4096	256	6	6	F	$(4, 4, 4, 4, 4)$	$(0, 0, 0, 0, 0)$	T	global	35.93	34.82	16.14

- Actual runtime very close to predictions
  - Same cluster based problem

PyMGRIT - model vs. runtime

N	N_P	L	K	cyc.	coarsening	# CF-relax.	nest. iter.	conv. crit.	PyMGRIT runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	T	local	19.99	19.53	18.01
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	F	local	18.87	18.43	18.02
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	T	local	17.69	17.17	15.09
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	F	local	16.98	16.66	15.09
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	T	global	24.66	23.91	18.06
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	F	global	23.28	21.94	18.01
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	T	global	25.08	24.04	15.07
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	F	global	24.24	23.15	15.08
4096	256	6	6	V	$(4,4,4,4,4)$	$(1,1,1,1,1)$	T	local	25.82	25.36	13.52
4096	256	6	6	F	$(4,4,4,4,4)$	$(0,0,0,0,0)$	T	local	26.06	25.43	16.13
4096	256	6	6	V	$(4,4,4,4,4)$	$(1,1,1,1,1)$	T	global	30.91	29.97	13.49
4096	256	6	6	F	$(4,4,4,4,4)$	$(0,0,0,0,0)$	T	global	35.93	34.82	16.14

- Actual runtime very close to predictions
  - Same cluster based problem
- Model approaches the theoretical limit when local criterion is used

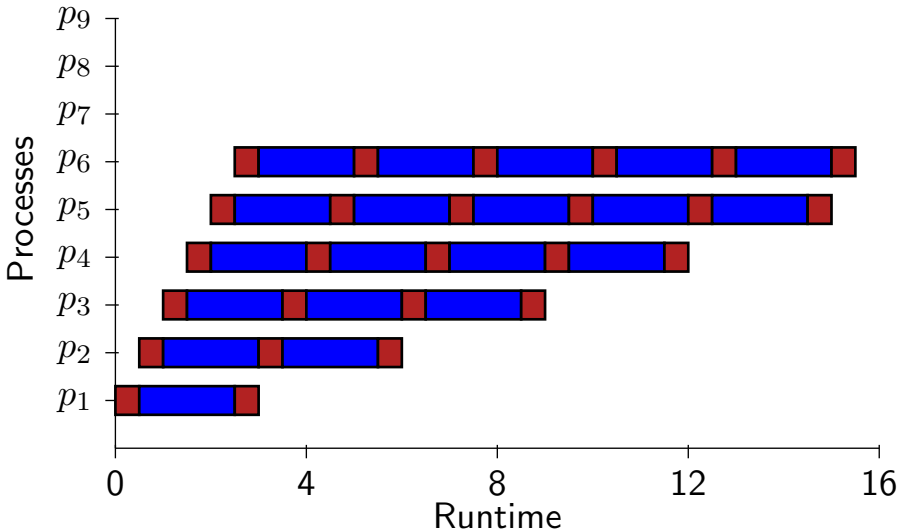
PyMGRIT - model vs. runtime

N	N_P	L	K	cyc.	coarsening	# CF-relax.	nest. iter.	conv. crit.	PyMGRIT runtime	Model	L.b. ($N_P = \infty$)
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	T	local	19.99	19.53	18.01
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	F	local	18.87	18.43	18.02
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	T	local	17.69	17.17	15.09
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	F	local	16.98	16.66	15.09
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	T	global	24.66	23.91	18.06
4096	256	5	6	V	$(16,4,4,4)$	$(1,1,1,1)$	F	global	23.28	21.94	18.01
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	T	global	25.08	24.04	15.07
4096	256	5	6	F	$(16,4,4,4)$	$(0,0,0,0)$	F	global	24.24	23.15	15.08
4096	256	6	6	V	$(4,4,4,4,4)$	$(1,1,1,1,1)$	T	local	25.82	25.36	13.52
4096	256	6	6	F	$(4,4,4,4,4)$	$(0,0,0,0,0)$	T	local	26.06	25.43	16.13
4096	256	6	6	V	$(4,4,4,4,4)$	$(1,1,1,1,1)$	T	global	30.91	29.97	13.49
4096	256	6	6	F	$(4,4,4,4,4)$	$(0,0,0,0,0)$	T	global	35.93	34.82	16.14

- Actual runtime very close to predictions
  - Same cluster based problem
- Model approaches the theoretical limit when local criterion is used
- Preceding nested iteration theoretically with zero cost

Outlook: Use task graph for scheduling

- Parareal example: $N = 6$, $K = 5$, Cost $\mathcal{G}$: 0.5 s, Cost $\mathcal{F}$: 2 s

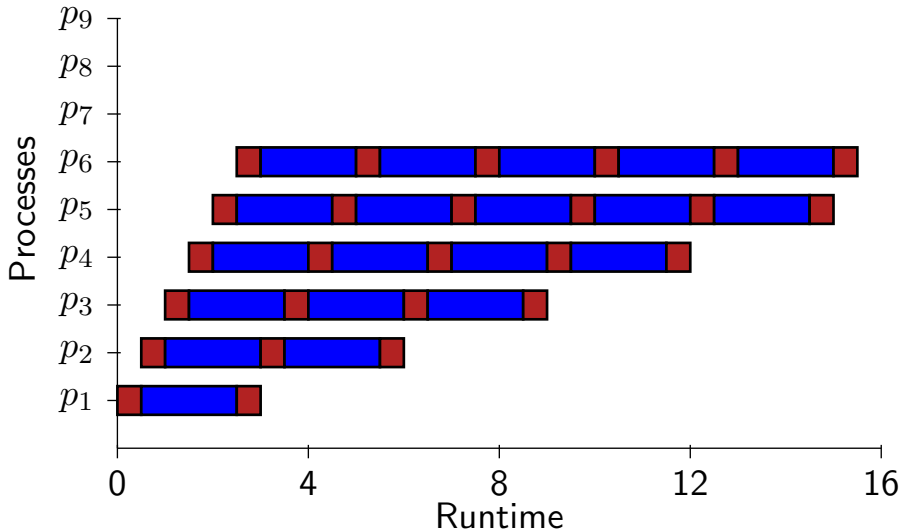


Block-by-block:

- $N_P = 6 = N$
- Runtime: 15.5 s

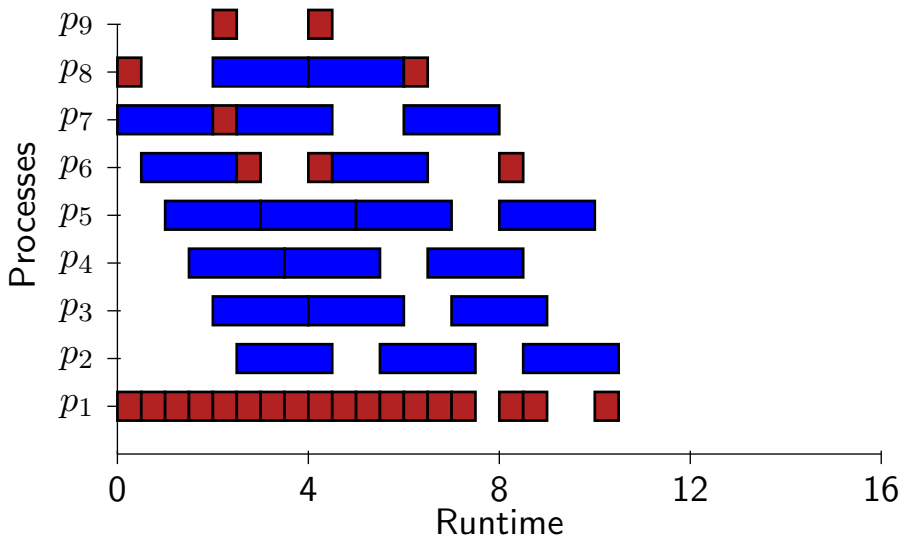
Outlook: Use task graph for scheduling

- Parareal example: $N = 6$, $K = 5$, Cost $\mathcal{G}$: 0.5 s, Cost $\mathcal{F}$: 2 s



Block-by-block:

- $N_P = 6 = N$
- Runtime: 15.5 s

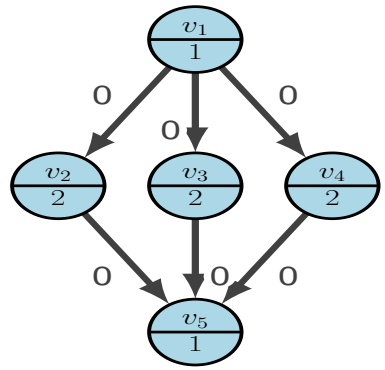


"Other" approach:

- $N_P = 9$
- Runtime: 10.5 s

Conclusions

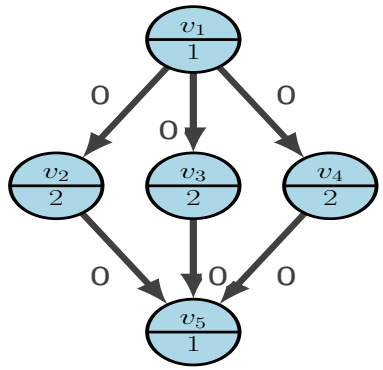
- Task-based analysis of PinT methods



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, Switzerland. This work is supported by the BMBF (project TIME-X; grant no. 16HPC046K).

Conclusions

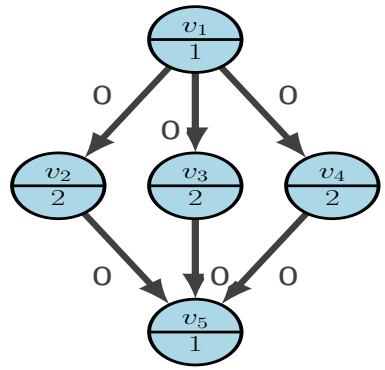
- Task-based analysis of PinT methods
- Covers large parameter space of PinT methods



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, Switzerland. This work is supported by the BMBF (project TIME-X; grant no. 16HPC046K).

Conclusions

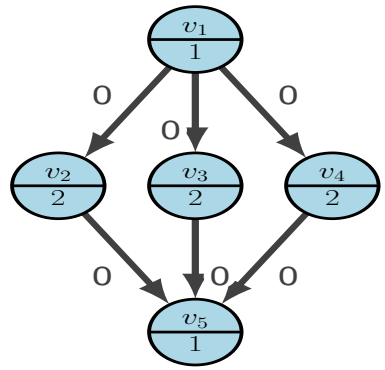
- Task-based analysis of PinT methods
- Covers large parameter space of PinT methods
- Theoretical lower runtime bound based on graph



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, Switzerland. This work is supported by the BMBF (project TIME-X; grant no. 16HPC046K).

Conclusions

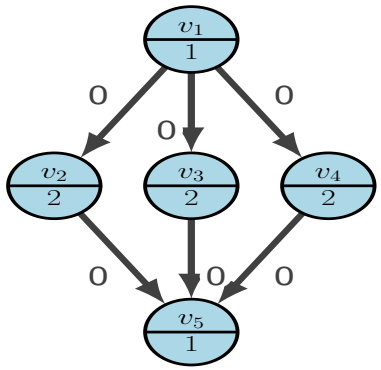
- Task-based analysis of PinT methods
- Covers large parameter space of PinT methods
- Theoretical lower runtime bound based on graph
- Runtime prediction using schedules



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, Switzerland. This work is supported by the BMBF (project TIME-X; grant no. 16HPC046K).

Conclusions

- Task-based analysis of PinT methods
- Covers large parameter space of PinT methods
- Theoretical lower runtime bound based on graph
- Runtime prediction using schedules
- Future work:
  - Load balancing strategies
  - Combine model with convergence predictions



SPONSORED BY THE
Federal Ministry of Education and Research

TIME-X

EuroHPC Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, Switzerland. This work is supported by the BMBF (project TIME-X; grant no. 16HPC046K).