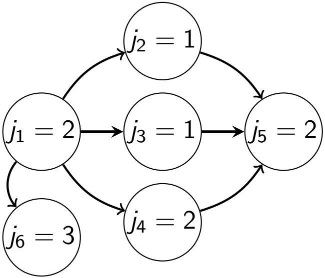
Reservation-Based Federated Scheduling for Parallel Real-Time Tasks

Niklas Ueter[1], Georg von der Brüggen[1], Jing Li[2], Jian-Jia Chen[1], Kunal Agrawal[3]

[1]**TU Dortmund University, Germany**
[2]**New Jersey Institute of Technology, USA**
[3]**Washington University in St. Louis**

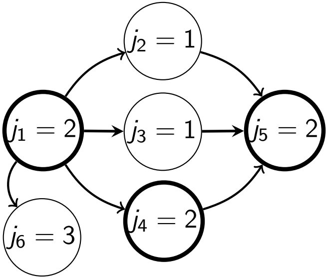
Table of Contents

Directed Acyclic Graph (DAG) Task Model



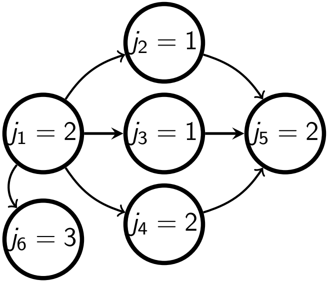
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$



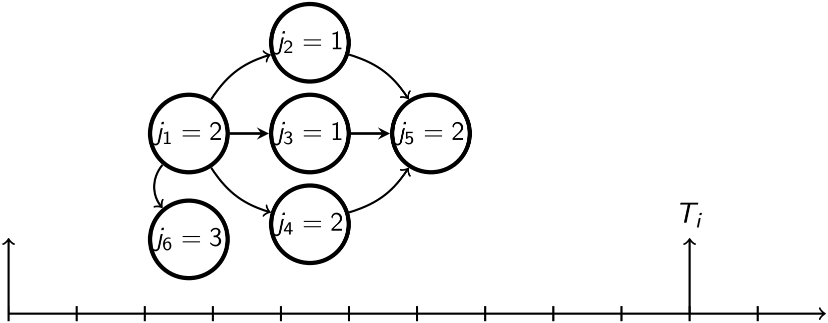
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$
- Cumulative worst-case execution time $vol(G_i)$



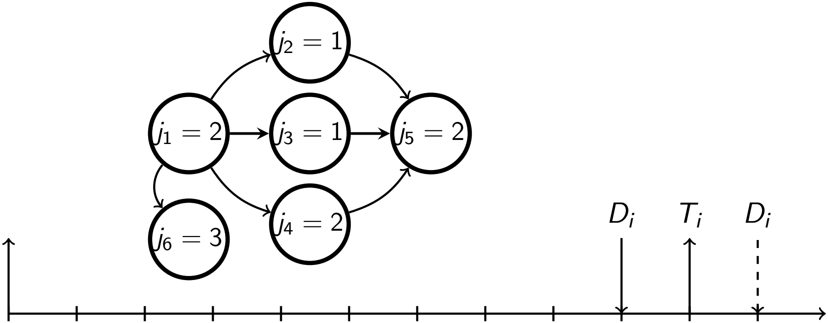
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$
- Cumulative worst-case execution time $vol(G_i)$
- Period T_i



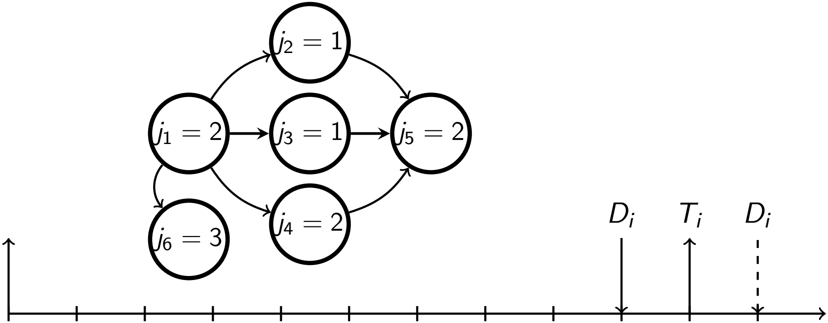
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$
- Cumulative worst-case execution time $vol(G_i)$
- Period T_i
- Deadline D_i



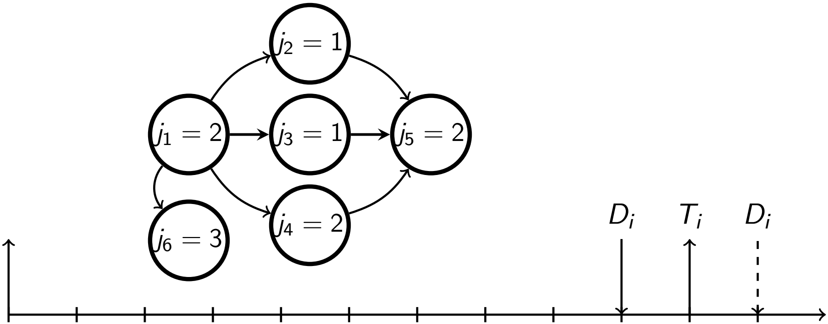
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$
- Cumulative worst-case execution time $vol(G_i)$
- Period T_i
- Deadline D_i



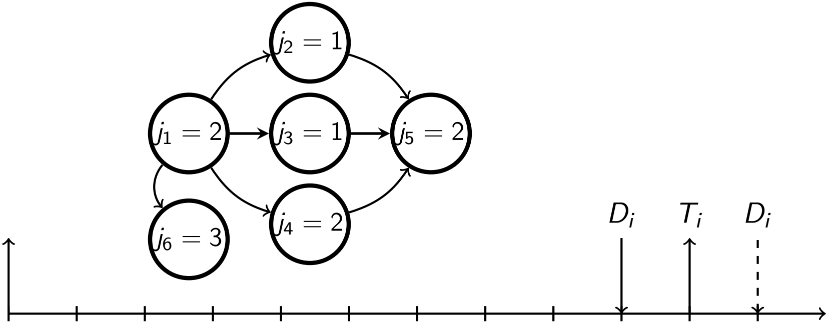
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$
- Cumulative worst-case execution time $vol(G_i)$
- Period T_i
- Deadline D_i
- Utilization $U_i = \frac{vol(G_i)}{T_i}$



Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $len(G_i)$
- Cumulative worst-case execution time $vol(G_i)$
- Period T_i
- Deadline D_i
- Utilization $U_i = \frac{vol(G_i)}{T_i}$
- Density $\delta_i = \frac{vol(G_i)}{\min\{D_i, T_i\}}$



System Model and Problem Definition

- Set $\mathbf{T}$ of sporadic arbitrary-deadline DAG tasks

System Model and Problem Definition

- Set **T** of sporadic arbitrary-deadline DAG tasks
- Set of **M** identical processors

System Model and Problem Definition

- Set **T** of sporadic arbitrary-deadline DAG tasks
- Set of **M** identical processors
- Schedule all tasks such that no task misses it's deadline

Federated Scheduling

- DAG task set is partitioned into **light** and **heavy** tasks

Federated Scheduling

- DAG task set is partitioned into **light** and **heavy** tasks
- Each **heavy** task is allocated to processors exclusively

Federated Scheduling

- DAG task set is partitioned into **light** and **heavy** tasks
- Each **heavy** task is allocated to processors exclusively
- All **light** tasks are scheduled on the remaining processors

Federated Scheduling

- DAG task set is partitioned into **light** and **heavy** tasks
- Each **heavy** task is allocated to processors exclusively
- All **light** tasks are scheduled on the remaining processors

We only focus on heavy tasks

Problem with Federated Scheduling?

$$m = \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)}$$

Problem with Federated Scheduling?

$$m = \left\lceil \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rceil$$

Problem with Federated Scheduling?

$$\left\lceil \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rceil - \left\lfloor \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rfloor$$

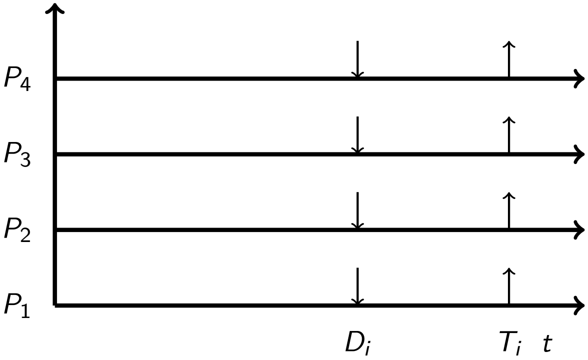
Problem with Federated Scheduling?

$$\underbrace{\left\lceil \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rceil - \left\lfloor \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rfloor}_{\epsilon}$$

Semi-Federated Scheduling [Jiang - RTSS 2017]

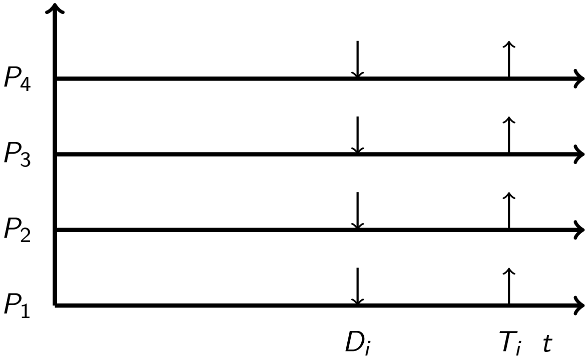
$$\left\lfloor \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rfloor + \epsilon$$

Problems with Federated & Semi-Federated Scheduling



Problems with Federated & Semi-Federated Scheduling

$\max\left\{0, \frac{T_i - D_i}{T_i}\right\}$ of processor capacity is at least wasted



Reservation-Based Federated Scheduling

Reserve **sufficient resources** for heavy tasks

but **not** necessarily **entire processors**

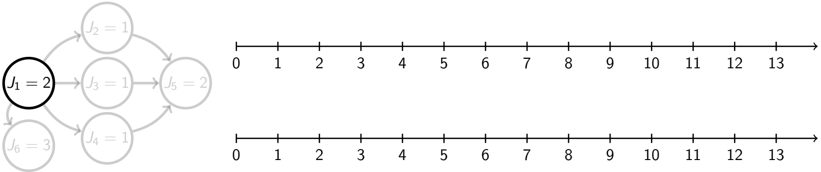
Reservation-Based Federated Scheduling

1. Specify a set of reservation servers for each heavy DAG task
2. Schedule the set of reservation servers

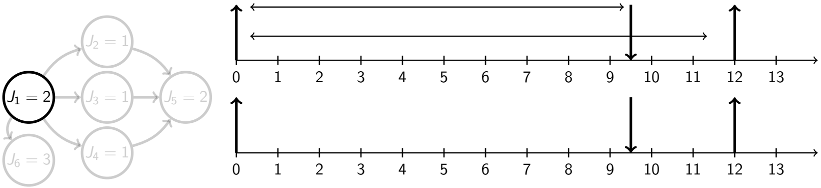
Reservation-Based Federated Scheduling

1. Specify a set of reservation servers for each heavy DAG task
2. Schedule the set of reservation servers
3. Schedule each DAG task within it's reservation servers

Reservation Servers

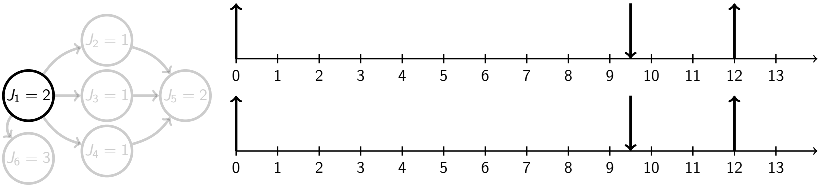


Reservation Servers



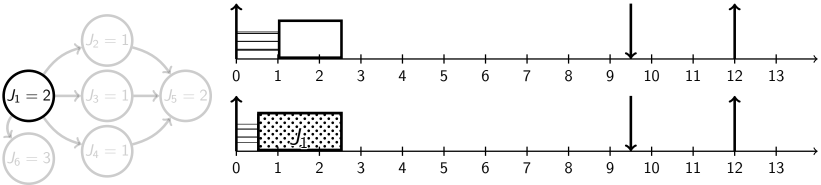
1. Sporadic arbitrary-deadline DAG task is released

Reservation Servers



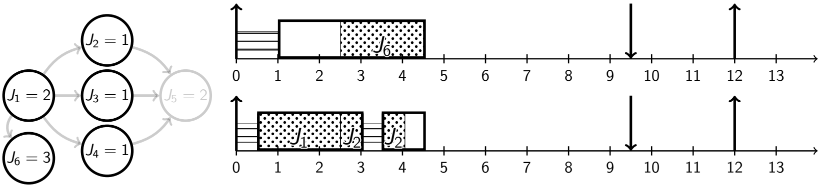
1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy

Reservation Servers



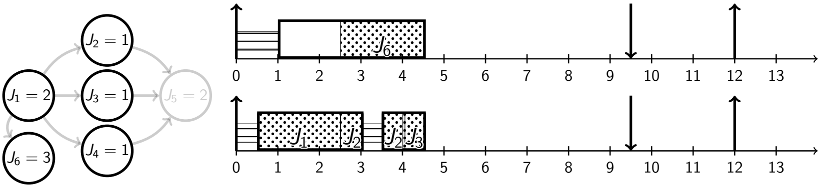
1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled

Reservation Servers



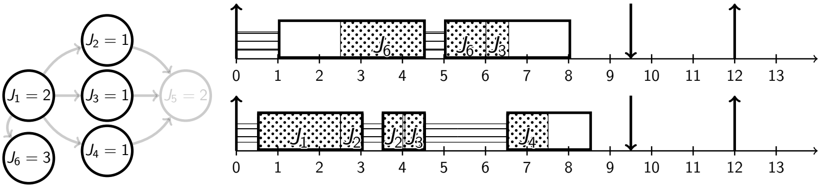
1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled

Reservation Servers



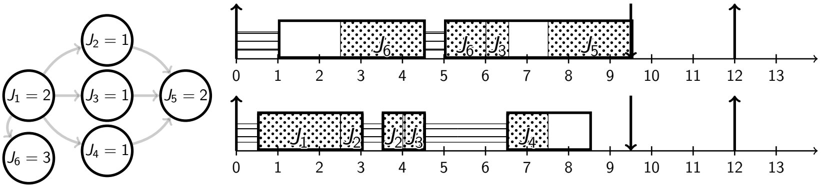
1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled

Reservation Servers



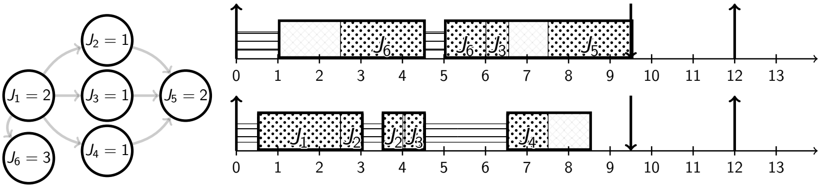
1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled

Reservation Servers



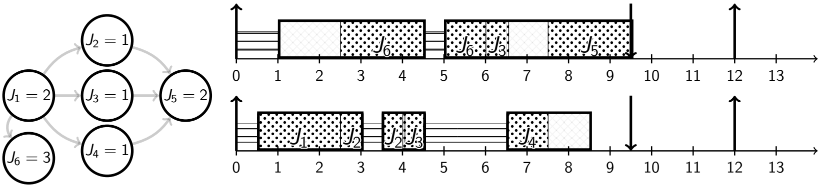
1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled

Reservation Servers



1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled

Reservation Servers



1. Sporadic arbitrary-deadline DAG task is released
2. Reservations are *released* with the DAG task and scheduled according to some policy
3. DAG task is serviced whenever a reservation is scheduled
4. How much reservation is required?

How to calculate the reservations?

$$vol(G_i) + len(G_i) \cdot (m_i - 1) \leq \sum_{j=1}^{m_i} E_j = \sum_{j=1}^{m_i} \gamma_j \cdot len(G_i)$$

How to calculate the reservations?

$$\mathbf{vol(G_i)} + len(G_i) \cdot (m_i - 1) \leq \sum_{j=1}^{m_i} \gamma_j \cdot len(G_i)$$

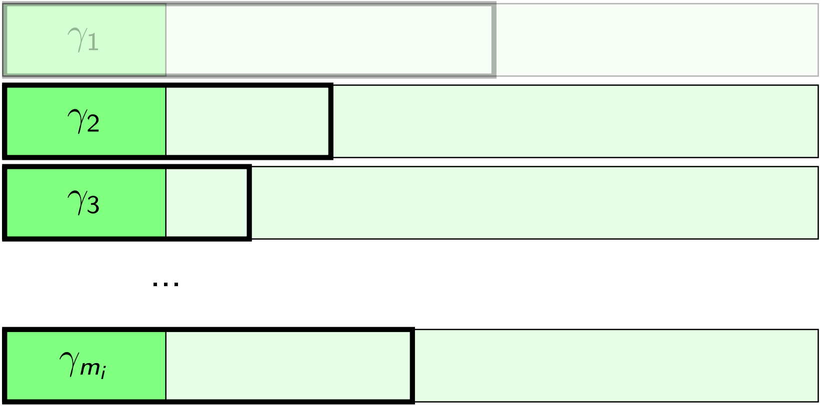
How to calculate the reservations?

$$vol(G_i) + \underbrace{\mathbf{len(G_i) \cdot (m_i - 1)}}_{critical\ \ path\ \ interference} \leq \sum_{j=1}^{m_i} \gamma_j \cdot len(G_i)$$

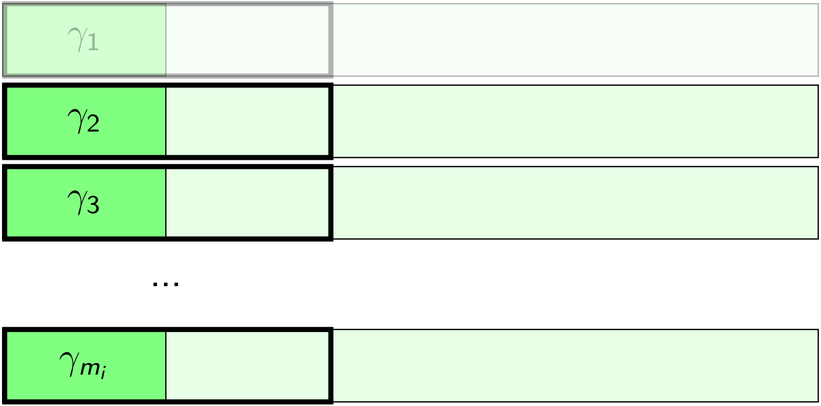
How to calculate the reservations?

$$vol(G_i) + len(G_i) \cdot (m_i - 1) \leq \sum_{j=1}^{m_i} \gamma_j \cdot len(G_i)$$

How to calculate the reservations?



How to calculate the reservations?



What does equal inflation give?

$$\underbrace{m_i \cdot \gamma_i \cdot len(G_i)}_{cumulative\ \ reservation} \leq \left(1 + \frac{1}{\gamma_i - 1}\right) \cdot vol(G_i)$$

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Maximal Inflation Reservations)

Set γ_j of each task to the maximal value.

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Maximal Inflation Reservations)

Set γ_j of each task to the maximal value.

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Maximal Inflation Reservations)

Set γ_j of each task to the maximal value.

- Minimal number of in-parallel reservations

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Maximal Inflation Reservations)

Set γ_j of each task to the maximal value.

- Minimal number of in-parallel reservations
- Minimal sufficient cumulative amount of service

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Maximal Inflation Reservations)

Set γ_j of each task to the maximal value.

- Minimal number of in-parallel reservations
- Minimal sufficient cumulative amount of service
- $\mathit{service} \leq (1 + \frac{1}{\gamma_j - 1}) \cdot \mathit{vol}(G_i)$

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Maximal Inflation Reservations)

Set γ_j of each task to the maximal value.

- Minimal number of in-parallel reservations
- Minimal sufficient cumulative amount of service
- $\mathit{service} \leq (1 + \frac{1}{\gamma_j - 1}) \cdot \mathit{vol}(G_i)$

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Constant Inflation Reservations)

Set all γ_j of each task to the same (feasible) value γ

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Constant Inflation Reservations)

Set all γ_j of each task to the same (feasible) value γ

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Constant Inflation Reservations)

Set all γ_j of each task to the same (feasible) value γ

How to calculate systems of reservation servers?

- **Decoupled** Calculate the reservation servers based on insights
- **Coupled** Calculate the reservation servers based on the scheduling problem

Example (Decoupled: Constant Inflation Reservations)

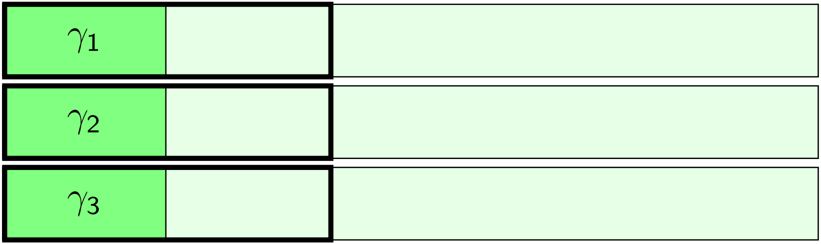
Set all γ_j of each task to the same (feasible) value γ

- Constant speedup factor of $3 + 2 \cdot \sqrt{2}$ (Global & Partitioned)

Coupled Algorithm: Split-On-Fail

1. M Processors and reservation servers for each task in $\mathbf{T}$

$$vol(G_i) + len(G_i) \cdot (\mathbf{3} - 1) \leq len(G_i) \cdot (\gamma_\mathbf{1} + \gamma_\mathbf{2} + \gamma_\mathbf{3})$$



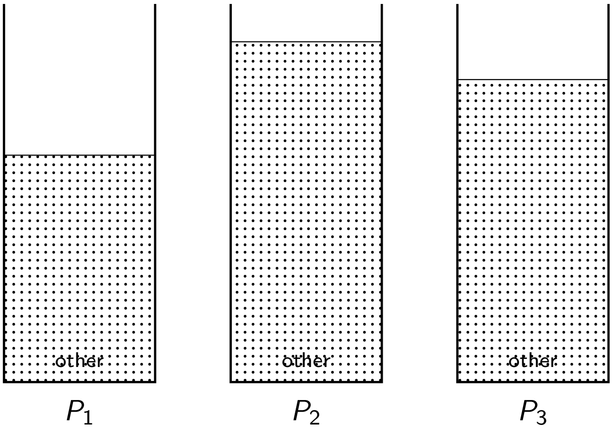
Coupled Algorithm: Split-On-Fail

1. M Processors and reservation servers for each task in $\mathbf{T}$
2. Try to partition reservation servers

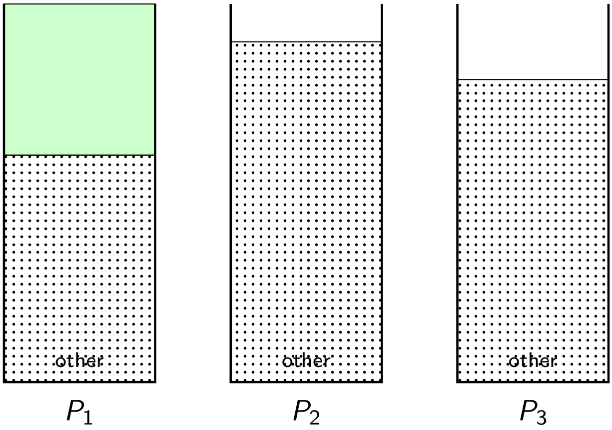
Coupled Algorithm: Split-On-Fail

1. M Processors and reservation servers for each task in $\mathbf{T}$
2. Try to partition reservation servers
3. Not schedulable? Increment number of reservations and decrease reservation budget

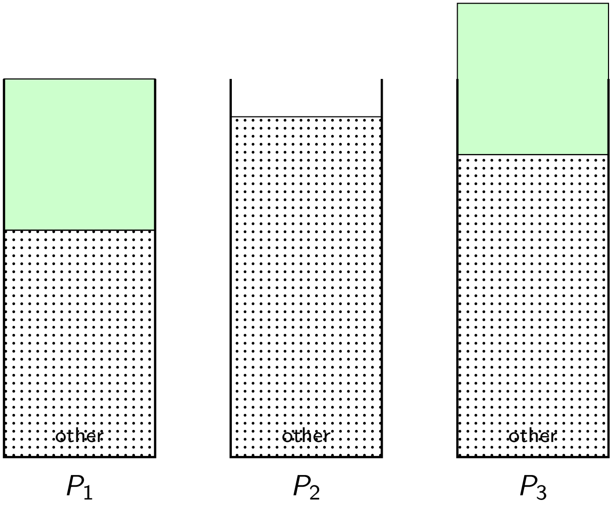
Coupled Algortithm: Split-On-Fail



Coupled Algortithm: Split-On-Fail

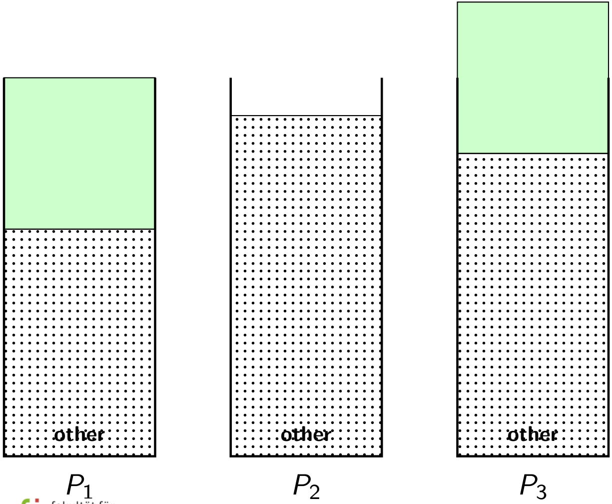


Coupled Algortithm: Split-On-Fail



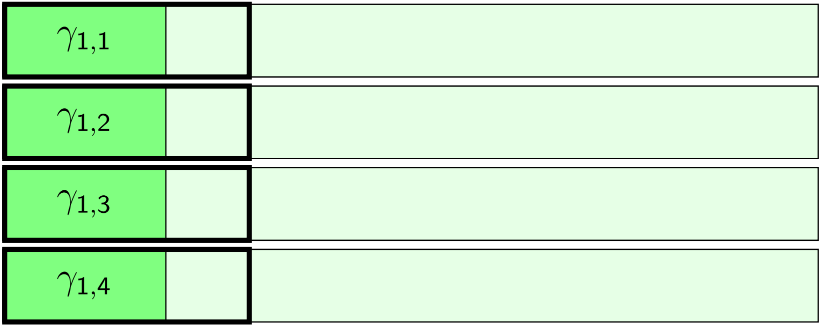
Coupled Algortithm: Split-On-Fail

Increment number of reservation servers and decrease budget

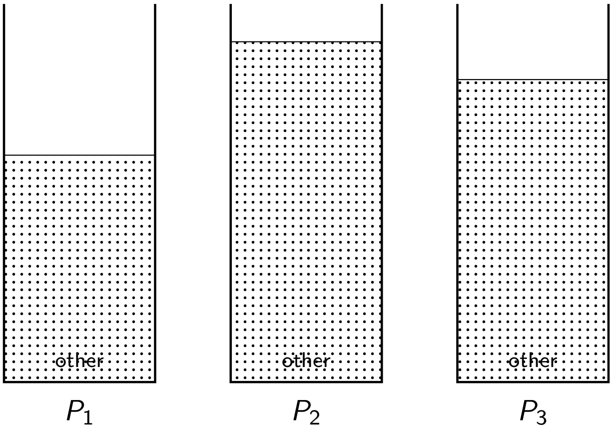


Coupled Algortithm: Split-On-Fail

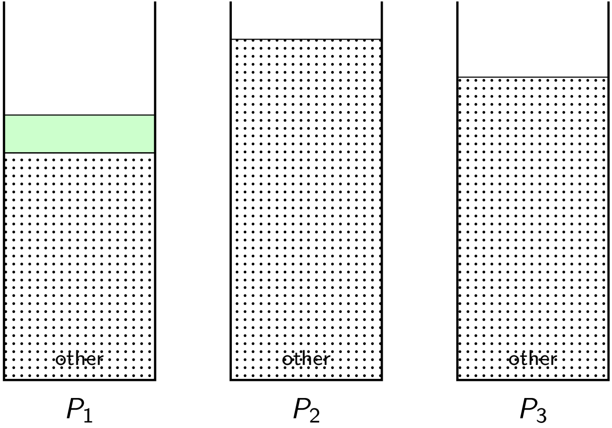
$$vol(G_i) + len(G_i) \cdot (\mathbf{4} - 1) \leq len(G_i) \cdot (\gamma_{\mathbf{1}} + \gamma_{\mathbf{2}} + \gamma_{\mathbf{3}} + \gamma_{\mathbf{4}})$$



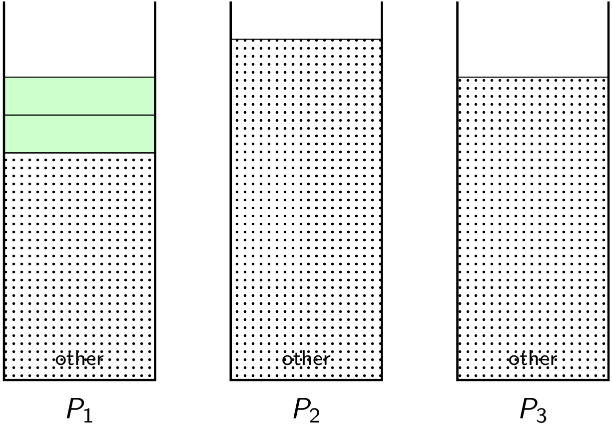
Coupled Algortithm: Split-On-Fail



Coupled Algortithm: Split-On-Fail

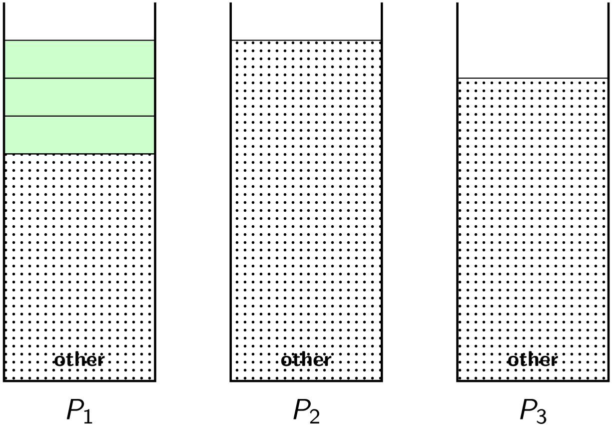


Coupled Algortithm: Split-On-Fail



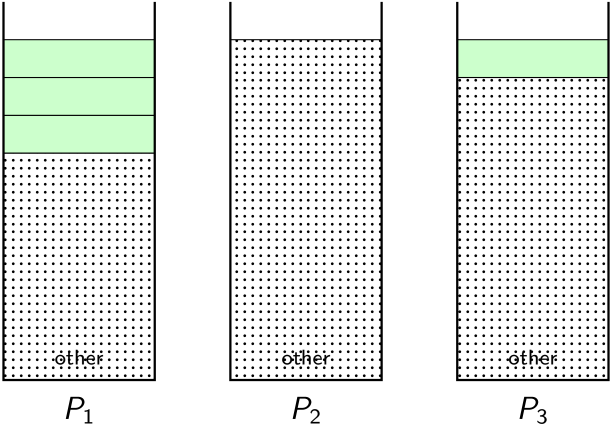
Coupled Algortithm: Split-On-Fail

Increment number of reservation servers and decrease budget



Coupled Algortithm: Split-On-Fail

$$vol(G_i) + len(G_i) \cdot (\mathbf{4} - 1) \leq len(G_i) \cdot (\gamma_{\mathbf{1}} + \gamma_{\mathbf{2}} + \gamma_{\mathbf{3}} + \gamma_{\mathbf{4}})$$



Experimental Results

Evaluations: DAG Task Set Generation

Parametric DAG Task Generation

- Task $\tau_i = (vol(G_i),\ len(G_i),\ D_i,\ T_i)$

Evaluations: DAG Task Set Generation

Parametric DAG Task Generation

- Task $\tau_i = (vol(G_i),\ len(G_i),\ D_i,\ T_i)$
- Randfixedsum for utilizations

Evaluations: DAG Task Set Generation

Parametric DAG Task Generation

- Task $\tau_i = (vol(G_i),\ len(G_i),\ D_i,\ T_i)$
- Randfixedsum for utilizations
- Uniformly distributed random variables: α, β, U_i, T_i:
  1. $0 < U_i \leq M$ (M is number of homogeneous processors)
  2. $0 < T_i \leq 100$
  3. $vol(G_i) = U_i \cdot T_i$
  4. $D_i = \alpha \cdot T_i$, and $len(G_i) = \beta \cdot D_i$

Evaluations: Experimental Setup

- 8, 16, and 32 identical processors systems

Evaluations: Experimental Setup

- 8, 16, and 32 identical processors systems
- 100 task sets with 20 DAG tasks each

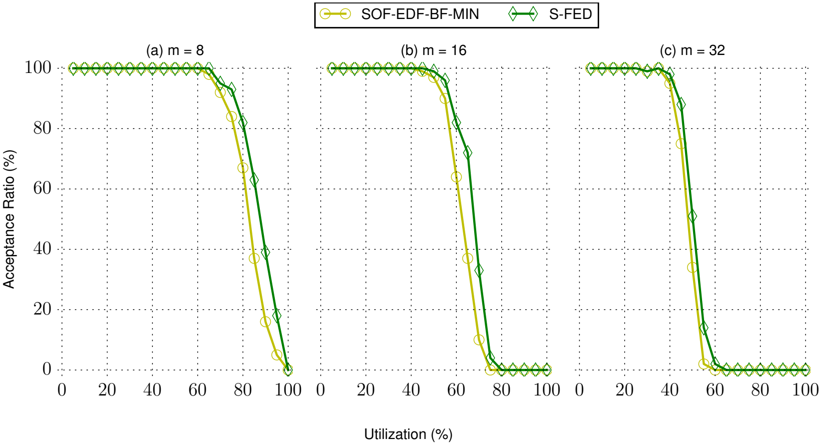
Evaluations: Experimental Setup

- 8, 16, and 32 identical processors systems
- 100 task sets with 20 DAG tasks each
- Set size: **large enough** to allow smaller individual utilizations
  $\Longrightarrow$ more options for the partitioning

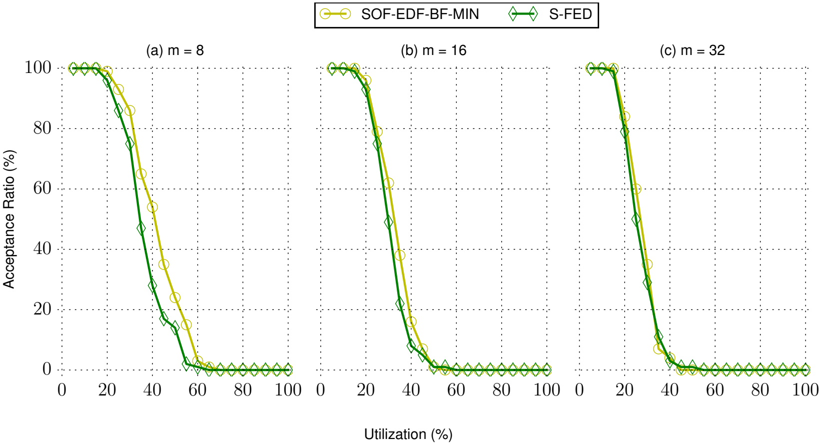
Evaluations: Experimental Setup

- 8, 16, and 32 identical processors systems
- 100 task sets with 20 DAG tasks each
- Set size: **large enough** to allow smaller individual utilizations
  $\Longrightarrow$ more options for the partitioning
- But small enough to be **difficult enough** to partition

Implicit-deadline DAG task sets



Constrained-deadline DAG task sets



Conclusion

- Use reservations to service an arbitrary-deadline DAG task

Conclusion

- Use reservations to service an arbitrary-deadline DAG task
- Generalizes Federated Scheduling

Conclusion

- Use reservations to service an arbitrary-deadline DAG task
- Generalizes Federated Scheduling
- Allows to separate the scheduling problem:
  1. Any scheduling of reservation servers (sporadic arbitrary-deadline task model)
  2. List Scheduling of a DAG task within reservations

Conclusion

- Use reservations to service an arbitrary-deadline DAG task
- Generalizes Federated Scheduling
- Allows to separate the scheduling problem:
  1. Any scheduling of reservation servers (sporadic arbitrary-deadline task model)
  2. List Scheduling of a DAG task within reservations
- Constant speedup factor of $3 + 2 \cdot \sqrt{2}$ (Global & Partitioned) for arbitrary-deadline DAG task systems

Thank You! Questions?

Conclusion

- Use reservations to service an arbitrary-deadline DAG task
- Generalizes Federated Scheduling
- Allows to separate the scheduling problem:
  1. Any scheduling of reservation servers (sporadic arbitrary-deadline task model)
  2. List Scheduling of a DAG task within reservations
- Constant speedup factor of $3 + 2 \cdot \sqrt{2}$ (Global & Partitioned) for arbitrary-deadline DAG task systems
- Implementation exist

Thank You! Questions?