Reservation-Based Federated Scheduling for Parallel Real-Time Tasks

Niklas Ueter\textsuperscript{1}, Georg von der Brüggen\textsuperscript{1}, Jing Li\textsuperscript{2}, Jian-Jia Chen\textsuperscript{1}, Kunal Agrawal\textsuperscript{3}

\textsuperscript{1}TU Dortmund University, Germany
\textsuperscript{2}New Jersey Institute of Technology, USA
\textsuperscript{3}Washington University in St. Louis
Table of Contents

- Introduction
- Task and System Model
- Related Ideas and Concepts
- Reservation-Based Federated Scheduling
- Evaluations
Directed Acyclic Graph (DAG) Task Model

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

\[ j_1 = 2 \quad j_2 = 1 \quad j_3 = 1 \quad j_4 = 2 \quad j_5 = 2 \quad j_6 = 3 \]
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: $\text{len}(G_i)$
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: \( \text{len}(G_i) \)
- Cumulative worst-case execution time \( \text{vol}(G_i) \)

\[
\begin{align*}
  j_1 &= 2 \\
  j_2 &= 1 \\
  j_3 &= 1 \\
  j_4 &= 2 \\
  j_5 &= 2 \\
  j_6 &= 3
\end{align*}
\]
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: \( \text{len}(G_i) \)
- Cumulative worst-case execution time \( \text{vol}(G_i) \)
- Period \( T_i \)

\[ j_1 = 2 \rightarrow j_2 = 1 \rightarrow j_3 = 1 \rightarrow j_5 = 2 \rightarrow j_4 = 2 \rightarrow j_6 = 3 \]

\[ T_i \]
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: \( \text{len}(G_i) \)
- Cumulative worst-case execution time \( \text{vol}(G_i) \)
- Period \( T_i \)
- Deadline \( D_i \)
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: \( \text{len}(G_i) \)
- Cumulative worst-case execution time \( \text{vol}(G_i) \)
- Period \( T_i \)
- Deadline \( D_i \)
Directed Acyclic Graph (DAG) Task Model

- Critical-path length: \( \text{len}(G_i) \)
- Cumulative worst-case execution time \( \text{vol}(G_i) \)
- Period \( T_i \)
- Deadline \( D_i \)
- Utilization \( U_i = \frac{\text{vol}(G_i)}{T_i} \)
Directed Acyclic Graph (DAG) Task Model

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

- Set $T$ 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 its 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?

\[ m = \frac{\text{vol}(G_i) - \text{len}(G_i)}{\min\{D_i, T_i\} - \text{len}(G_i)} \]
Problem with Federated Scheduling?

\[ m = \left\lfloor \frac{\text{vol}(G_i) - \text{len}(G_i)}{\min\{D_i, T_i\} - \text{len}(G_i)} \right\rfloor \]
Problem with Federated Scheduling?

\[ \left[ \frac{\text{vol}(G_i) - \text{len}(G_i)}{\min \{D_i, T_i\} - \text{len}(G_i)} \right] - \left[ \frac{\text{vol}(G_i) - \text{len}(G_i)}{\min \{D_i, T_i\} - \text{len}(G_i)} \right] \]
Problem with Federated Scheduling?

\[
\epsilon = \left\lceil \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rceil - \left\lfloor \frac{vol(G_i) - len(G_i)}{\min\{D_i, T_i\} - len(G_i)} \right\rfloor
\]
\[
\left\lceil \frac{\text{vol}(G_i) - \text{len}(G_i)}{\min \{D_i, T_i\} - \text{len}(G_i)} \right\rceil + \epsilon
\]
Problems with Federated & Semi-Federated Scheduling

\[ \max \{0, T_i - D_i \} \]

\[
\begin{align*}
&\quad P_4 \\
&\quad P_3 \\
&\quad P_2 \\
&\quad P_1 \\
&\quad D_i \\
&\quad T_i \\
&\quad t
\end{align*}
\]
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

A sporadic arbitrary-deadline DAG task is released. Reservations are released with the DAG task and scheduled according to some policy. The DAG task is serviced whenever a reservation is scheduled. How much reservation is required?
Reservation Servers

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
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
Sporadic arbitrary-deadline DAG task is released

Reservations are released with the DAG task and scheduled according to some policy

DAG task is serviced whenever a reservation is scheduled
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
Sporadic arbitrary-deadline DAG task is released

Reservations are released with the DAG task and scheduled according to some policy

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
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?

\[ vol(G_i) + \text{len}(G_i) \cdot (m_i - 1) \leq \sum_{j=1}^{m_i} E_j = \sum_{j=1}^{m_i} \gamma_j \cdot \text{len}(G_i) \]
How to calculate the reservations?

\[
\text{vol}(G_i) + \text{len}(G_i) \cdot (m_i - 1) \leq \sum_{j=1}^{m_i} \gamma_j \cdot \text{len}(G_i)
\]
How to calculate the reservations?

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

\[ \gamma_1 \]

\[ \gamma_2 \]

\[ \gamma_3 \]

\[ \ldots \]

\[ \gamma_{m_i} \]
How to calculate the reservations?

<table>
<thead>
<tr>
<th>$\gamma_1$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma_2$</td>
<td></td>
</tr>
<tr>
<td>$\gamma_3$</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$\gamma_{m_i}$</td>
<td></td>
</tr>
</tbody>
</table>
What does equal inflation give?

\[ m_i \cdot \gamma_i \cdot \text{len}(G_i) \leq \left( 1 + \frac{1}{\gamma_i - 1} \right) \cdot \text{vol}(G_i) \]
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 $\gamma_j$ 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 $\gamma_j$ 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 $\gamma_j$ 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 $\gamma_j$ 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 $\gamma_j$ of each task to the maximal value:

- Minimal number of in-parallel reservations
- Minimal sufficient cumulative amount of service
- $service \leq (1 + \frac{1}{\gamma_j-1}) \cdot vol(G_i)$
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 $\gamma_j$ of each task to the maximal value.
- Minimal number of in-parallel reservations
- Minimal sufficient cumulative amount of service
- $service \leq (1 + \frac{1}{\gamma_j - 1}) \cdot vol(G_i)$
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 $\gamma_j$ of each task to the same (feasible) value $\gamma$. 
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 $\gamma_j$ of each task to the same (feasible) value $\gamma$
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 $\gamma_j$ of each task to the same (feasible) value $\gamma$
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 $\gamma_j$ of each task to the same (feasible) value $\gamma$

- 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 $T$

$$vol(G_i) + len(G_i) \cdot (3 - 1) \leq len(G_i) \cdot (\gamma_1 + \gamma_2 + \gamma_3)$$

| $\gamma_1$ |  
| $\gamma_2$ |  
| $\gamma_3$ |  

TU Dortmund
**Coupled Algorithm: Split-On-Fail**

1. $M$ Processors and reservation servers for each task in $T$
2. Try to partition reservation servers
Coupled Algorithm: Split-On-Fail

1. \( M \) Processors and reservation servers for each task in \( T \)
2. Try to partition reservation servers
3. Not schedulable? Increment number of reservations and decrease reservation budget
Coupled Algorithm: Split-On-Fail

\[ P_1 \]

\[ P_2 \]

\[ P_3 \]
Coupled Algorithm: Split-On-Fail
Coupled Algorithm: Split-On-Fail

$P_1$  

$P_2$  

$P_3$  

\[ \text{other} \]
Coupled Algorithm: Split-On-Fail

Increment number of reservation servers and decrease budget

\[ P_1 \]
\[ \text{other} \]
\[ P_2 \]
\[ \text{other} \]
\[ P_3 \]
\[ \text{other} \]
Coupled Algorithm: Split-On-Fail

\[ \text{vol}(G_i) + \text{len}(G_i) \cdot (4 - 1) \leq \text{len}(G_i) \cdot (\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4) \]
Coupled Algorithm: Split-On-Fail

$P_1$  $P_2$  $P_3$
Coupled Algorithm: Split-On-Fail
Coupled Algorithm: Split-On-Fail

\[ P_1 \]
\[ P_2 \]
\[ P_3 \]
Coupled Algorithm: Split-On-Fail

Increment number of reservation servers and decrease budget

\[ P_1 \]
\[ P_2 \]
\[ P_3 \]

other

other

other
Coupled Algorithm: Split-On-Fail

$$vol(G_i) + len(G_i) \cdot (4 - 1) \leq len(G_i) \cdot (\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4)$$
Evaluations

Experimental Results
Evaluations: DAG Task Set Generation

Parametric DAG Task Generation

- Task $\tau_i = (\text{vol}(G_i), \text{len}(G_i), D_i, T_i)$
Evaluations: DAG Task Set Generation

Parametric DAG Task Generation

- Task $\tau_i = (\text{vol}(G_i), \text{len}(G_i), D_i, T_i)$
- Randfixedsum for utilizations
Evaluations: DAG Task Set Generation

Parametric DAG Task Generation

- Task $\tau_i = (\text{vol}(G_i), \text{len}(G_i), D_i, T_i)$
- Randfixedsum for utilizations
- Uniformly distributed random variables: $\alpha, \beta, U_i, T_i$:
  1. $0 < U_i \leq M$ ($M$ is number of homogeneous processors)
  2. $0 < T_i \leq 100$
  3. $\text{vol}(G_i) = U_i \cdot T_i$
  4. $D_i = \alpha \cdot T_i$, and $\text{len}(G_i) = \beta \cdot D_i$
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  
  ⇒ 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
  ⇒ more options for the partitioning
• But small enough to be **difficult enough** to partition
Implicit-deadline DAG task sets

(a) $m = 8$

(b) $m = 16$

(c) $m = 32$
Constrained-deadline DAG task sets

![Graphs showing acceptance ratio vs utilization for different m values](image)

- (a) m = 8
- (b) m = 16
- (c) m = 32

- SOF-EDF-BF-MIN
- S-FED

TU Dortmund
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

Constant speedup factor of \(3 + 2 \cdot \sqrt{2}\) for arbitrary-deadline DAG task systems

Implementation exist

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

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?