# Multiprocessor Scheduling of Elastic Tasks

Thursday, November 7, 2019

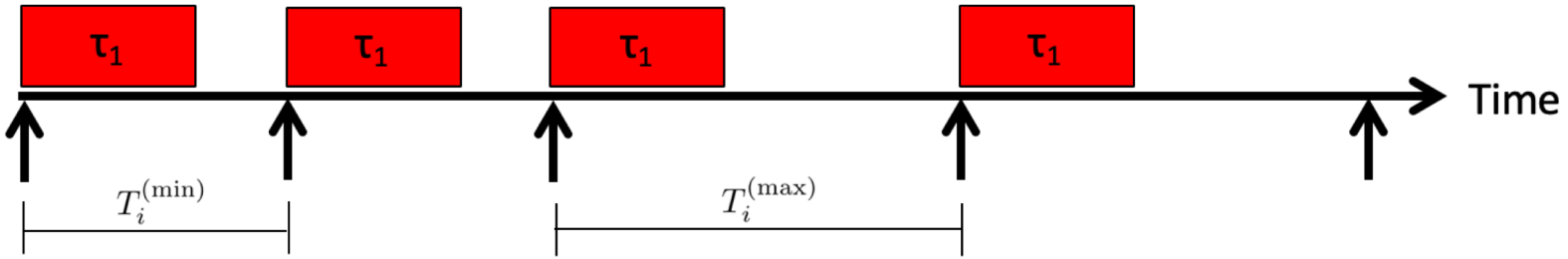
**James Orr**, Sanjoy Baruah

Ela

- E

del

- $T_i^{(\min)}$ - Minimum (Desired) Period, $\left(U_i^{(\max)} = \frac{C_i}{T_i^{(\min)}}\right)$
- $T_i^{(\max)}$ - Maximum (Acceptable) Period, $\left(U_i^{(\min)} = \frac{C_i}{T_i^{(\max)}}\right)$
- $T_i$ – Current Period $T_i^{(\min)} \leq T_i \leq T_i^{(\max)}$
- $C_i$ – Computation Time
- $E_i$ – Elastic Coefficient
  - » Higher $E_i$ implies a more elastic task




G. Buttazzo, G. Lipari, and L. Abeni
"Elastic Task Model for Adaptive Rate Control," RTSS 1998

$F$

# Elastic Scheduling of Real-Time Tasks

- Buttazzo et al. provide an iterative algorithm that increases periods from $T_i^{(\min)}$ proportionally to $E_i$ (as far as $T_i^{(\max)}$ ) for a uniprocessor

$$\lambda = \left(\frac{U_i^{(max)} - U_i}{E_i}\right) = \left(\frac{U_j^{(max)} - U_j}{E_j}\right)$$

$$U_i = U_i^{(\max)} - \lambda E_i ;$$

- However, each task can only be stretched to $U_i^{(\min)}$

$$U_i = max\left(U_i^{(\max)} - \lambda E_i, U_i^{(\min)}\right)$$

# Multiprocessor Scheduling of Elastic Tasks

$$\lambda = \left(\frac{U_i^{(max)} - U_i}{E_i}\right) = \left(\frac{U_j^{(max)} - U_j}{E_j}\right)$$

- Idea: preserve semantics of Buttazzo's algorithm
  - » Find smallest value of $\lambda$ such that all tasks are schedulable
  - » $\lambda$ must be in the range [0, Φ] where Φ is the maximum value among tasks of the equation $\left(\frac{U_i^{(max)} - U_i^{(min)}}{E_i}\right)$
  - » $\lambda$ =0 implies all tasks at $U_i^{(\max)}$
  - » $\lambda$ =Φ implies all tasks at $U_i^{(\min)}$

# Experiments to Validate the Approach

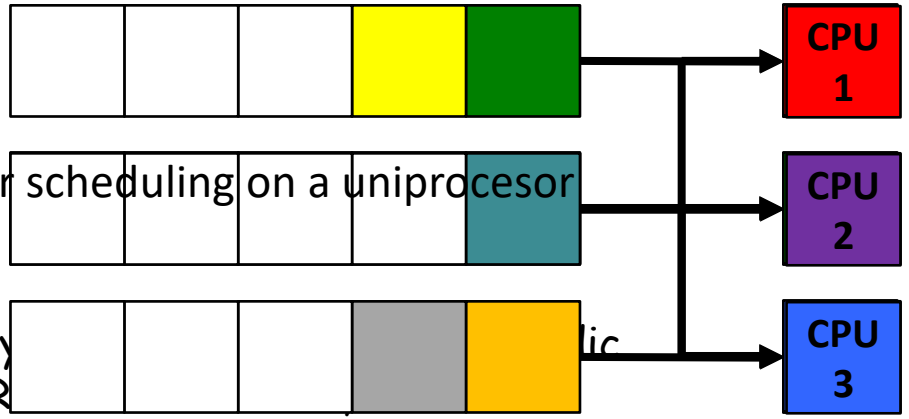
- Attempt to schedule (by simulation) task sets via 4 existing scheduling algorithms (smallest $\lambda$ wins)
  - **Fluid** – theoretical algorithm in which tasks can use a partial processor, can use full processor capacity
  - **Global EDF** – jobs scheduled where available
  - **PriD** – generalization of Global EDF tasks with highest $U_i$ get dedicated processors
  - **Partitioned** – jobs always on the same processor

Schedulable under Buttazzo's algorithm for scheduling on a uniprocesor

$$\sum_{\tau_i \in \Gamma} U_i \leq m - (m-1) \times \max_{\tau_i \in \Gamma} \{U_i\}$$

Can actually partition tasks in polynomial time

J. Goossens, S. Funk, and S. Baruah "Priority [illegible] Task Systems On Multiprocessors" R[illegible]



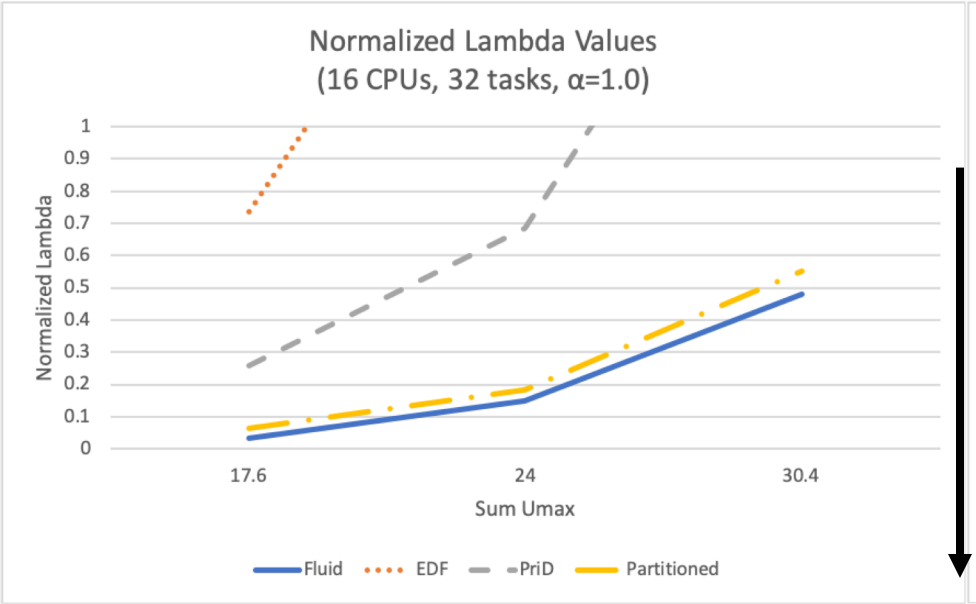
# Experimental Details

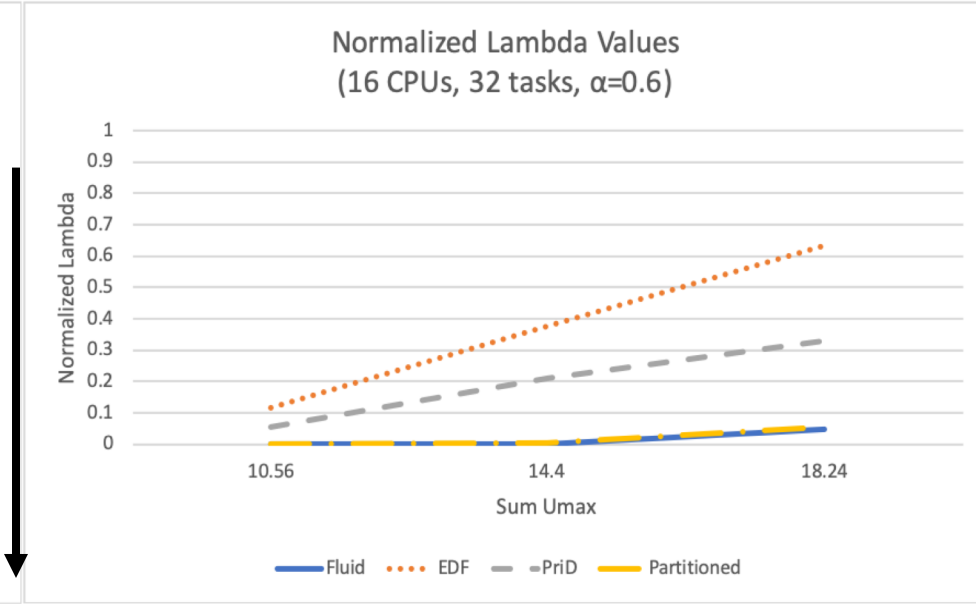
- For each scheduling algorithm iterate $\lambda$ over range [0, Φ] to find smallest value such that the taskset is schedulable
- Generate 1000 task sets per data point
  - m={8, 16, 32} CPUs
  - n={2m, 2.5m, 3m, 4m, 8m} tasks
  - Largest possible task size α={0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
  - $\left(\sum_i U_i^{(\max)}\right)$ ={1.1mα, 1.5mα, 1.9mα}

# Summary of Results

- Monotonic ordering of schedulability
  - Fluid > Partitioned > PriD > Global EDF
- Additional observations
  - Schedulability increases as $\alpha$ decreases
  - Schedulability increases as number of CPUs decreases
  - Schedulability (generally) increases as tasks/CPU increases

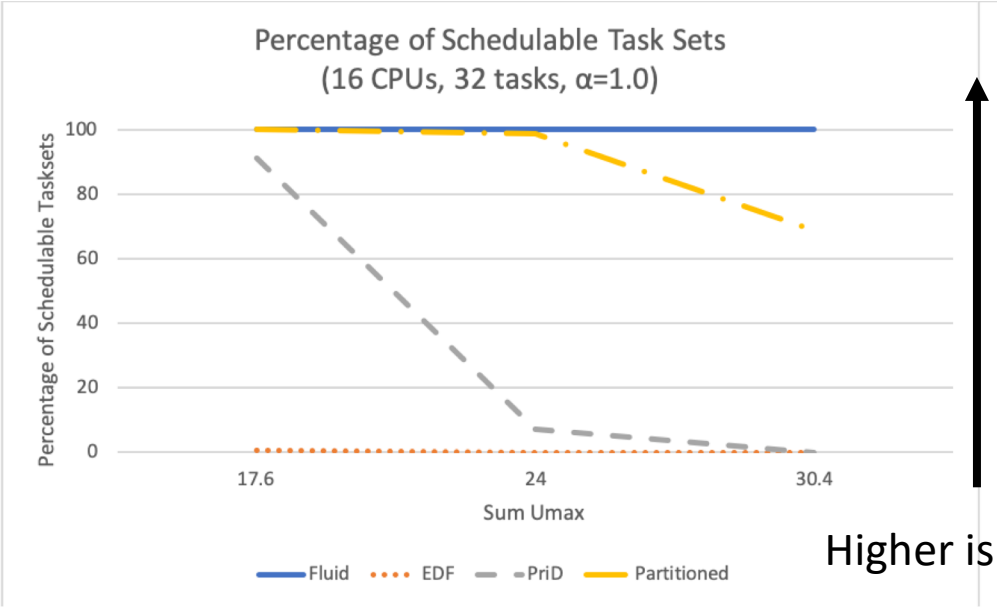
# Schedulability increases as α decreases



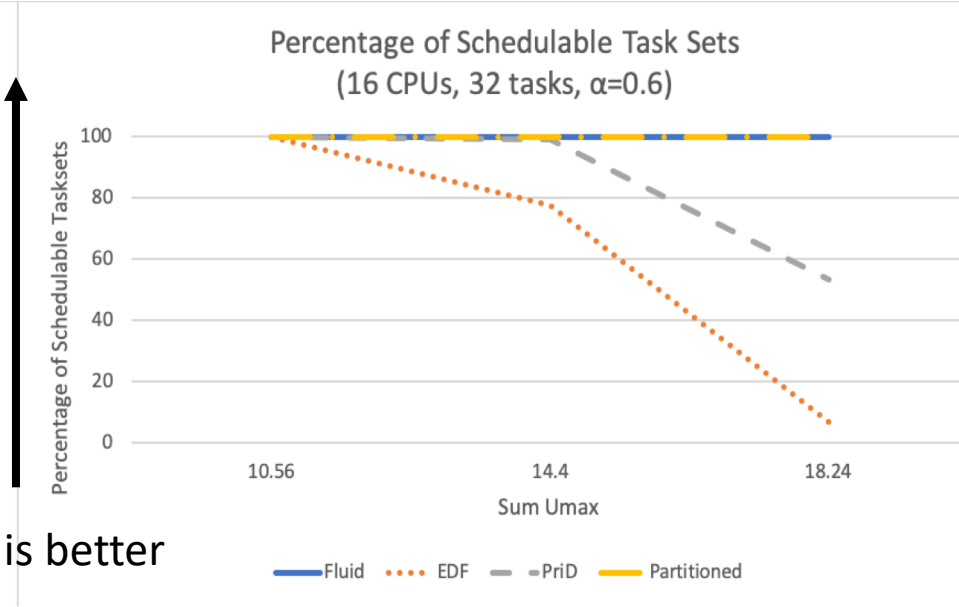




Lower is better

# Schedulability increases as α decreases

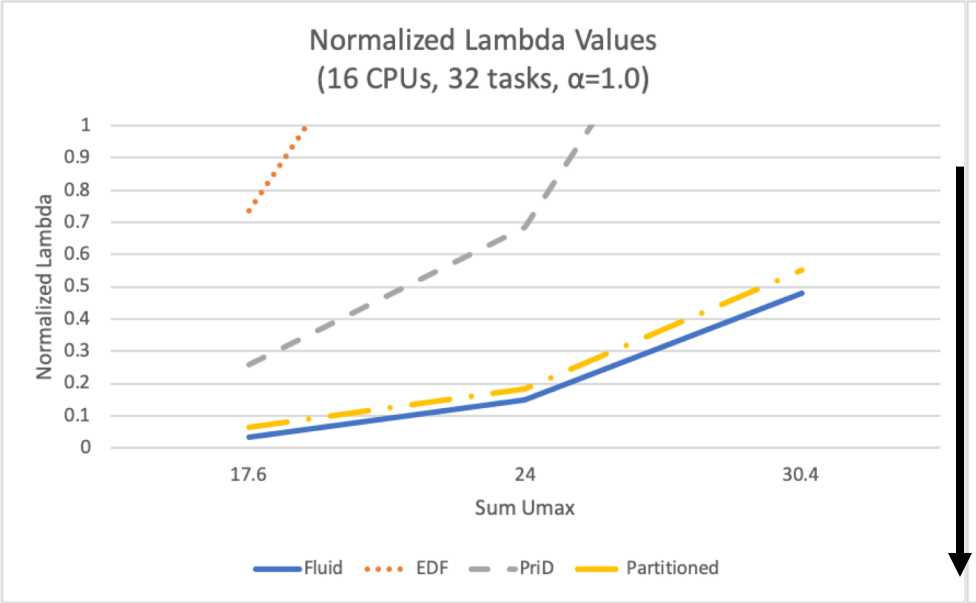


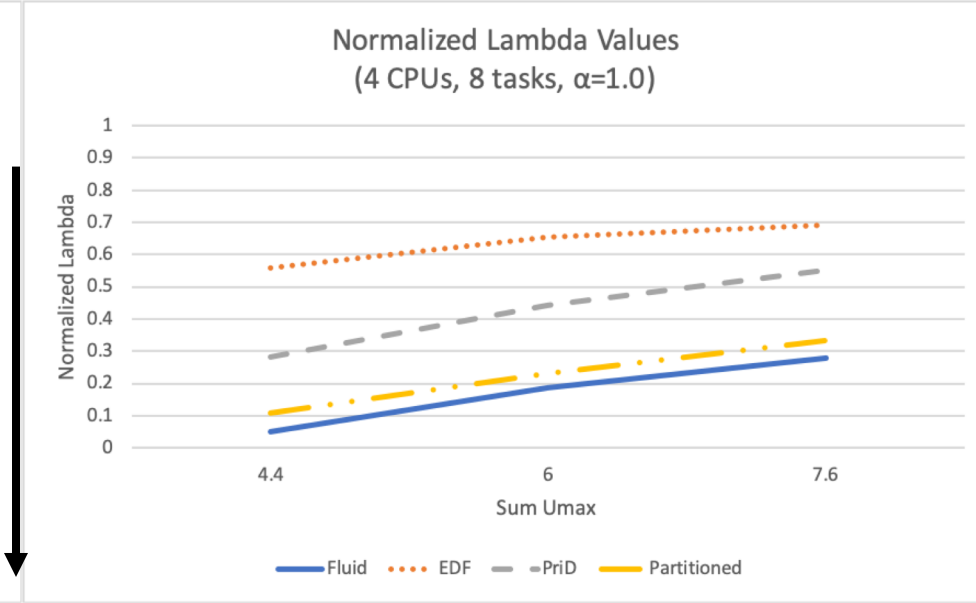

Higher is better



# Schedulability increases as number of CPUs decreases

(but tasks/CPU stay the same)

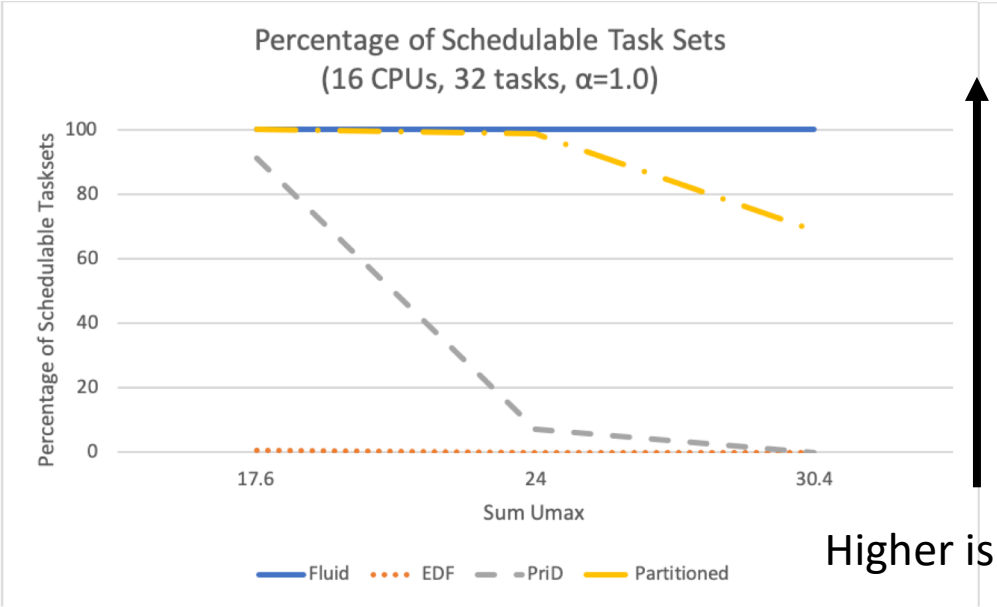


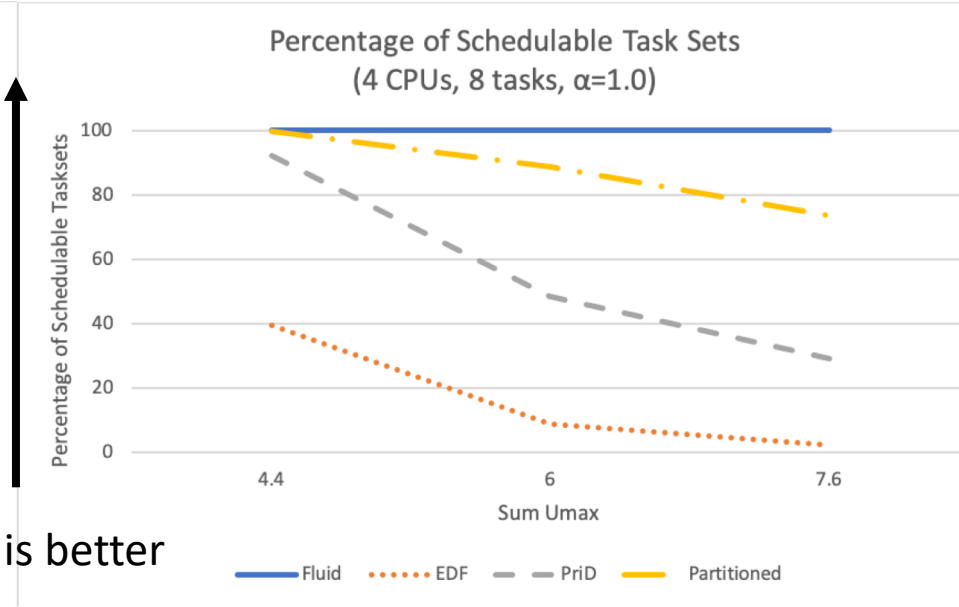




Lower is better

# Schedulability increases as number of CPUs decreases

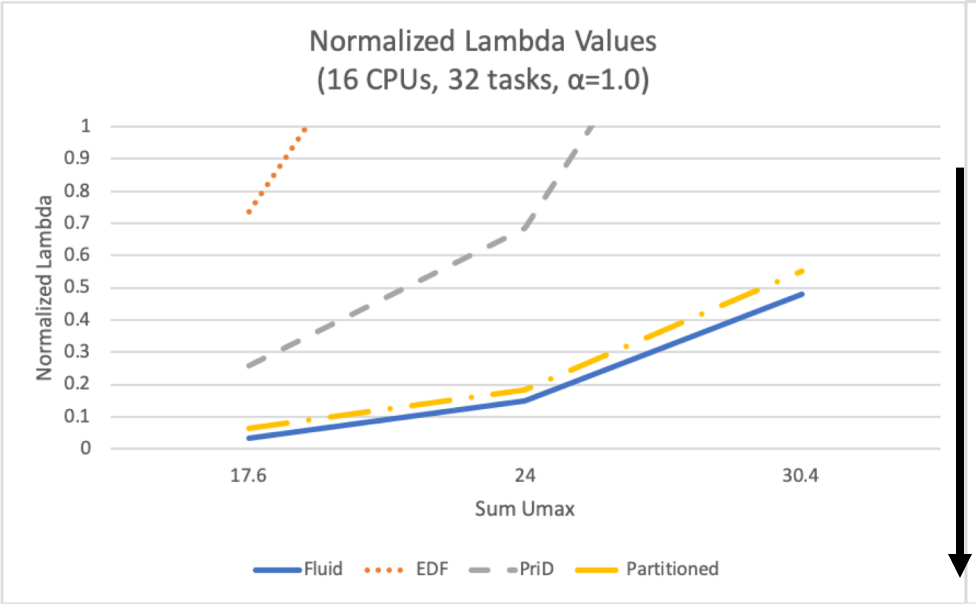
(but tasks/CPU stay the same)

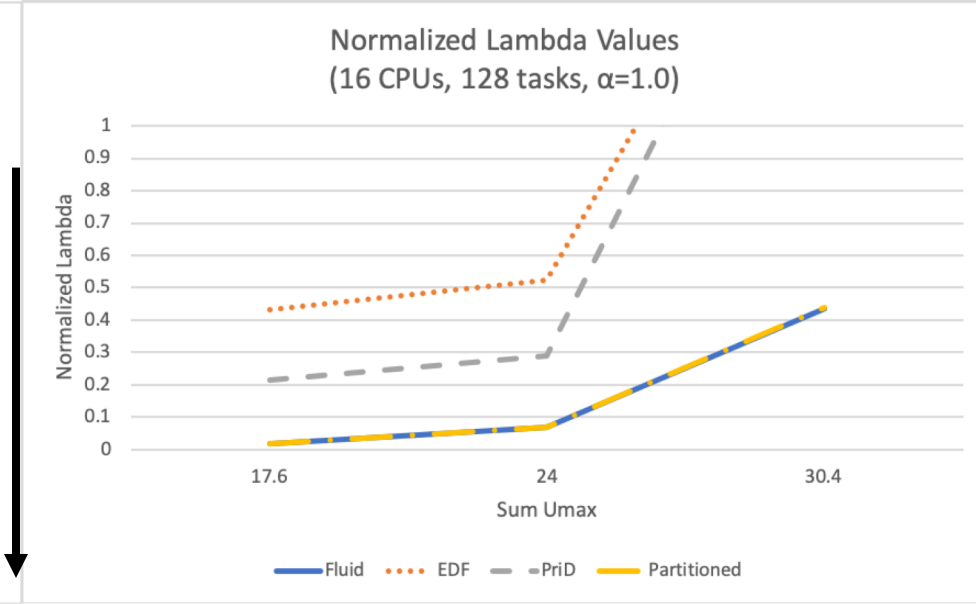



Higher is better

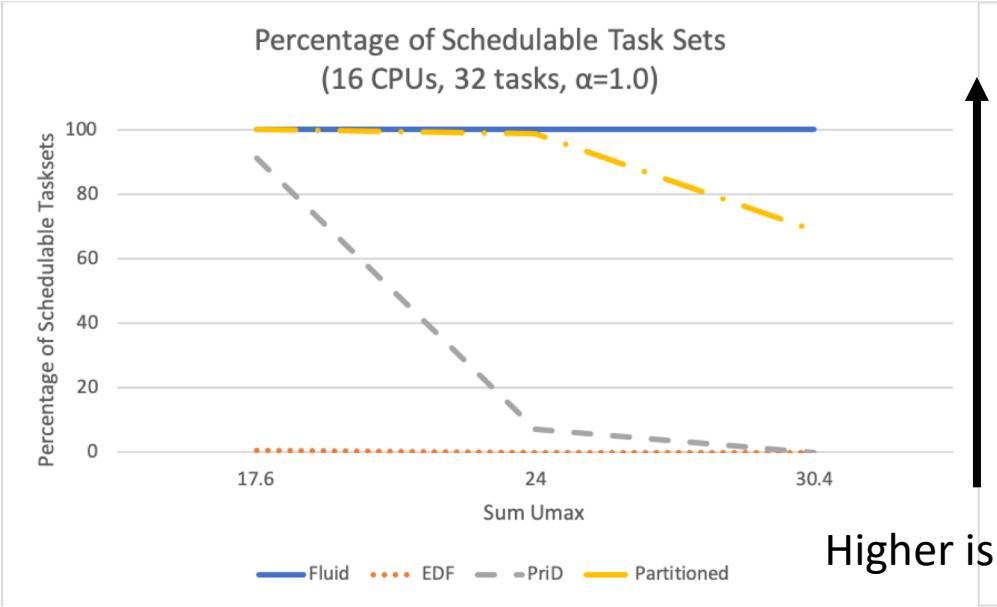
# Schedulability (generally) increases as tasks/CPU increases

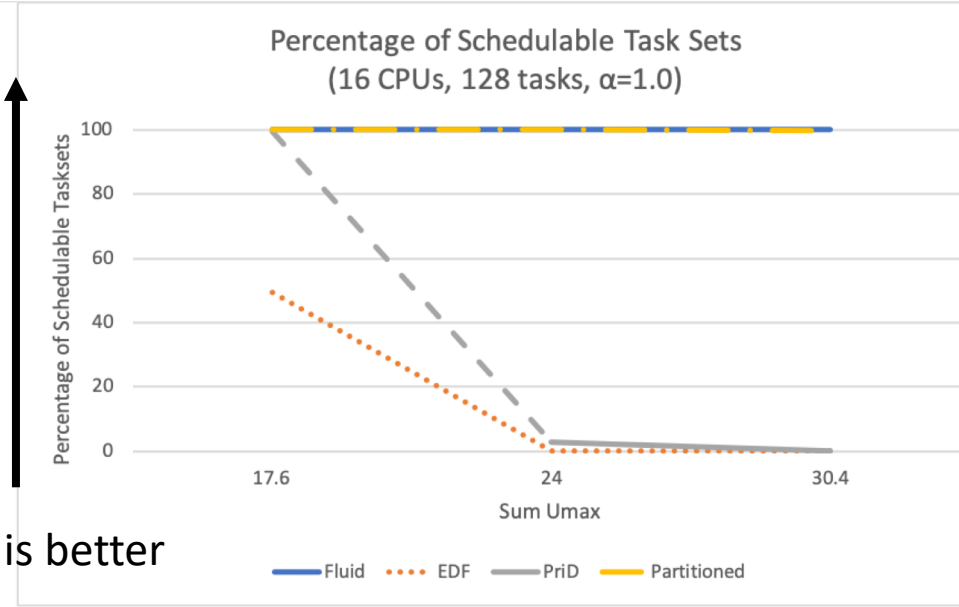



Lower is better

# Schedulability (generally) increases as tasks/CPU increases




Higher is better

# Summary of Results

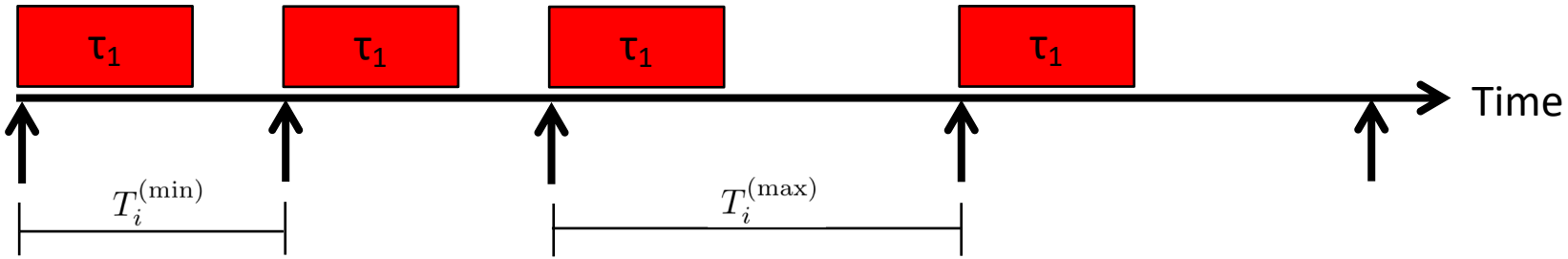
- Monotonic ordering of schedulability
  - Fluid > Partitioned > PriD > Global EDF
- Additional observations
  - Schedulability increases as $\alpha$ decreases
  - Schedulability increases as number of CPUs decreases
  - Schedulability (generally) increases as tasks/CPU increases

# Thank you. Questions?

Supported in part by NSF grant CNS-1911460

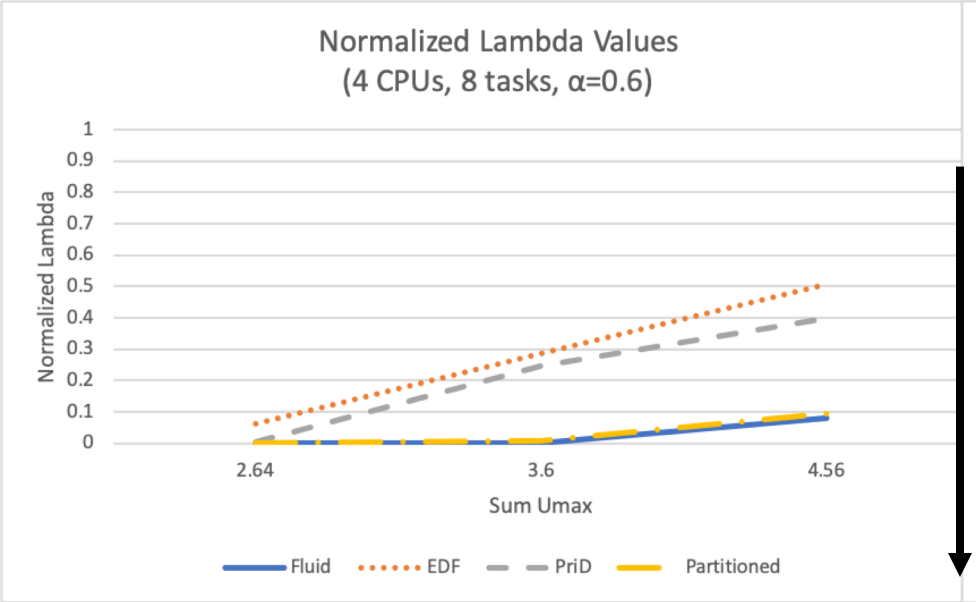
# Supplementary Slides

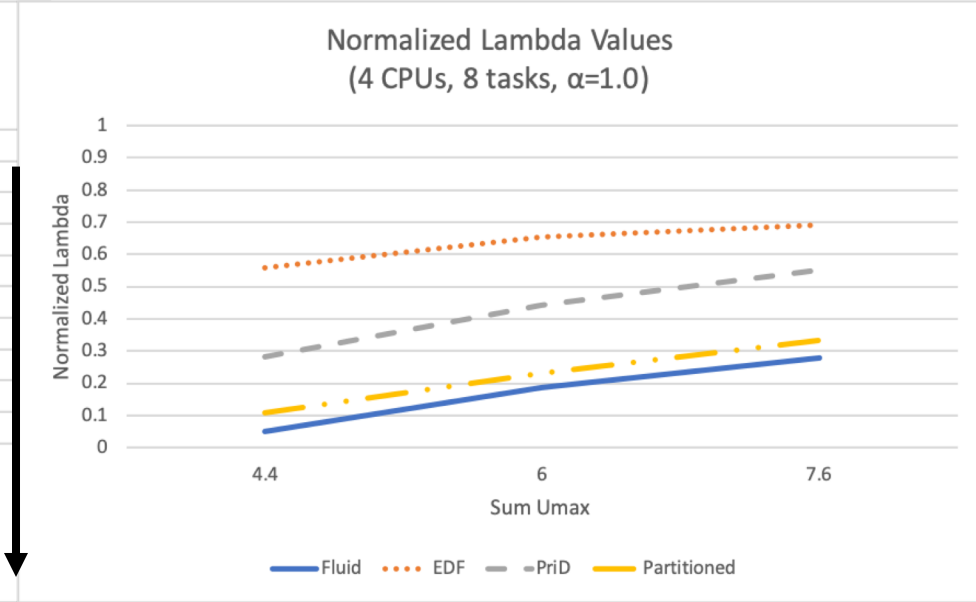
- $T_i^{(\min)}$ - Minimum (Desired) Period, $\left(U_i^{(\max)} = \frac{C_i}{T_i^{(\min)}}\right)$
- $T_i^{(\max)}$ - Maximum (Acceptable) Period, $\left(U_i^{(\min)} = \frac{C_i}{T_i^{(\max)}}\right)$
- $T_i$ – Current Period $T_i^{(\min)} \leq T_i \leq T_i^{(\max)}$
- $C_i$ – Computation Time
- $E_i$ – Elastic Coefficient
  - » Higher $E_i$ implies a more elastic task




G. Buttazzo, G. Lipari, and L. Abeni
"Elastic Task Model for Adaptive Rate Control," RTSS 1998

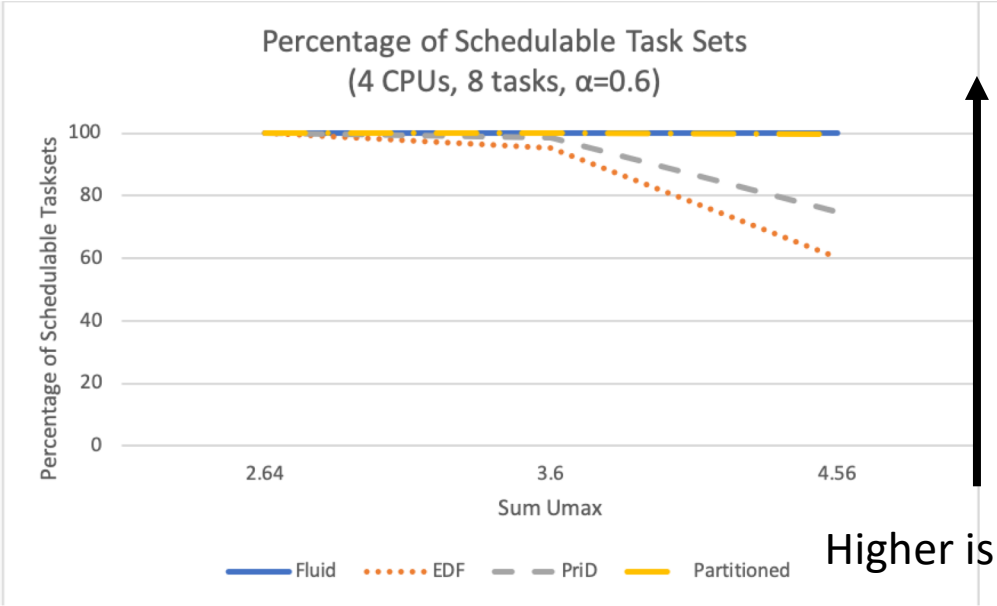
# Effects of changing α

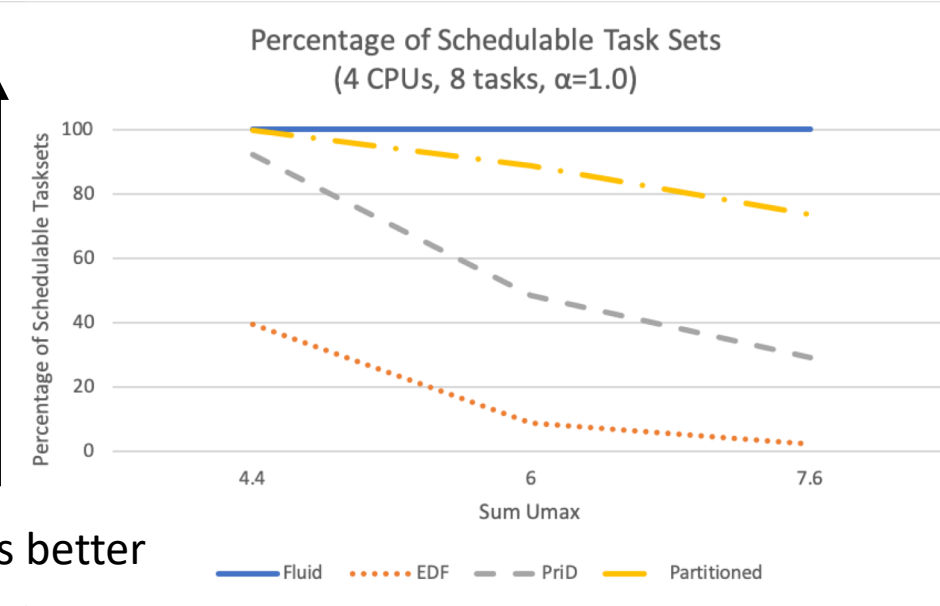



Lower is better

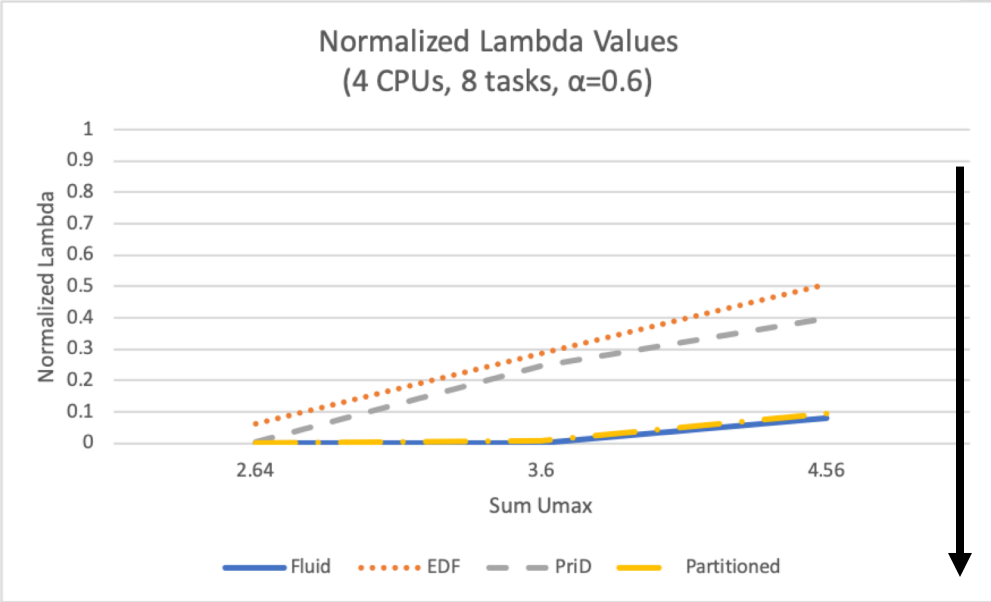
# Effects of changing α

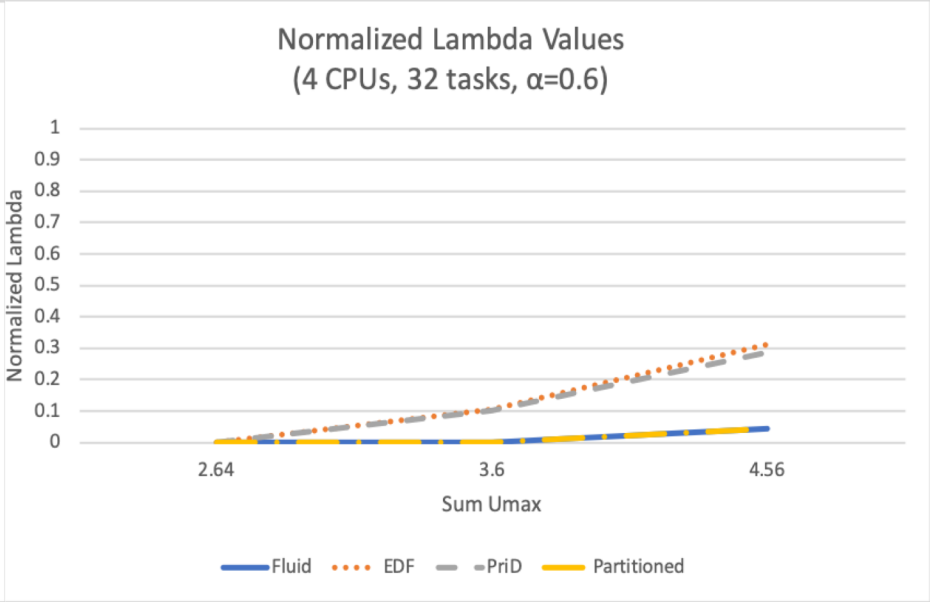



Higher is better

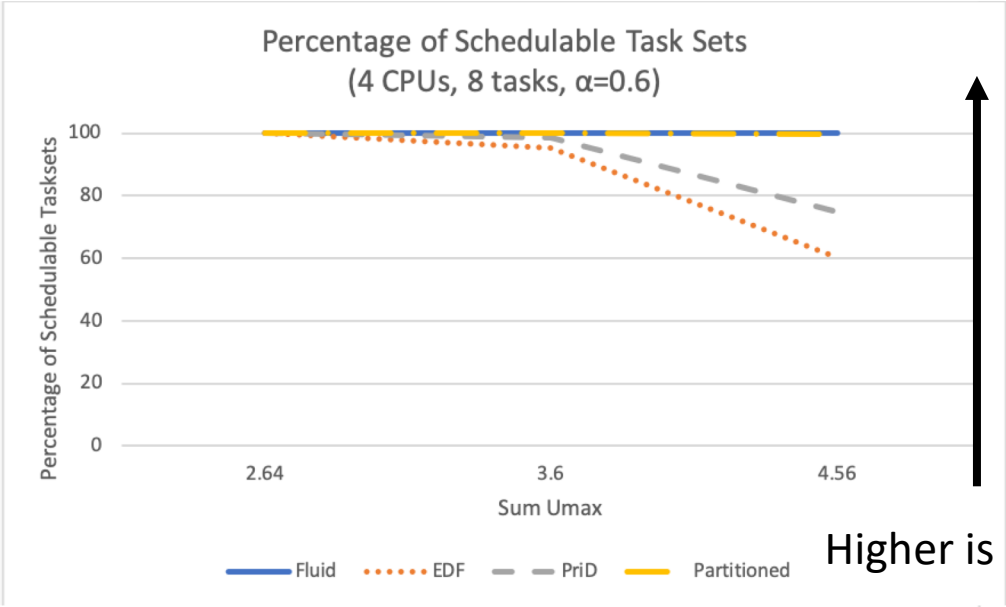
# Effects of changing no. tasks/CPU



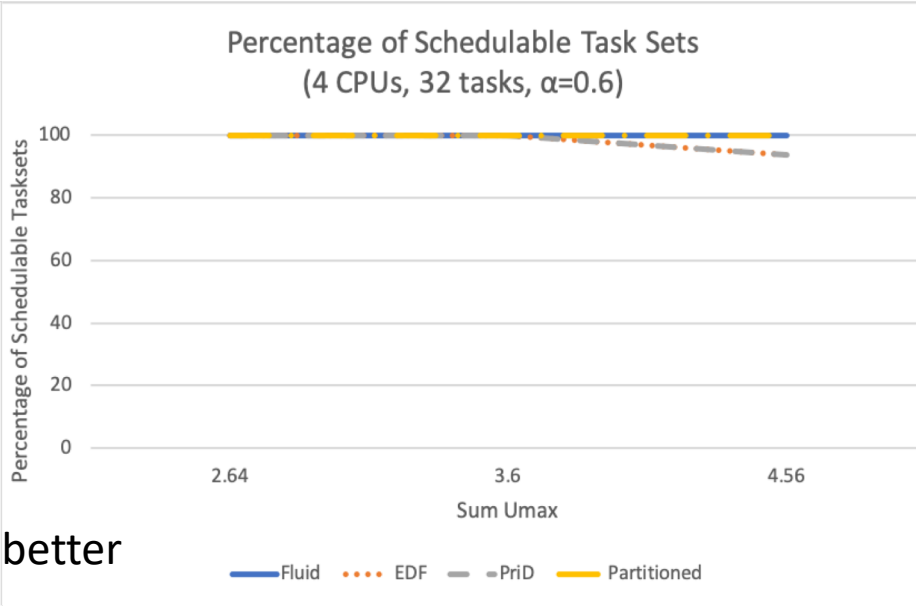




Lower is better

# Effects of changing no. tasks/CPU

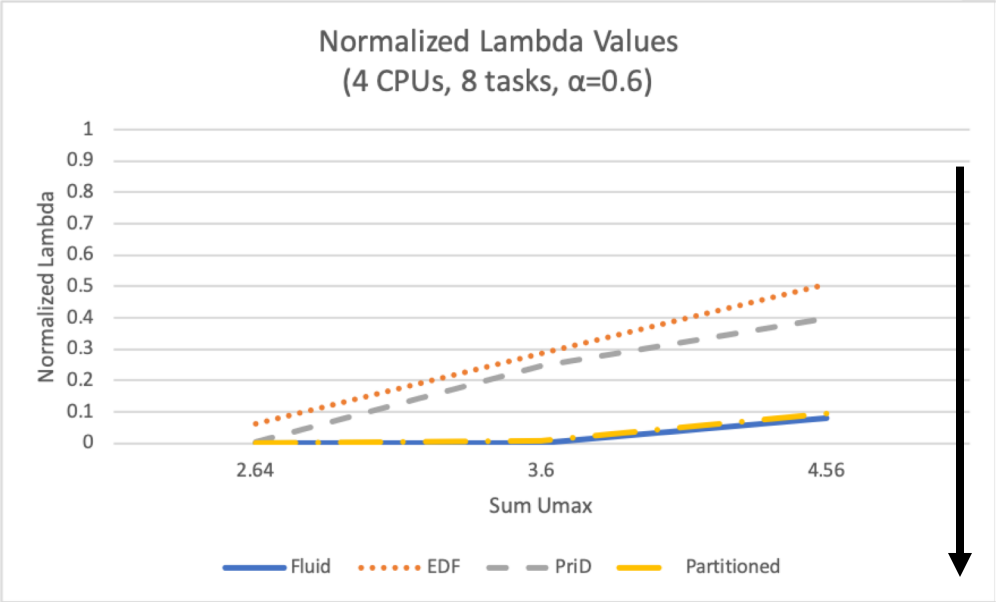


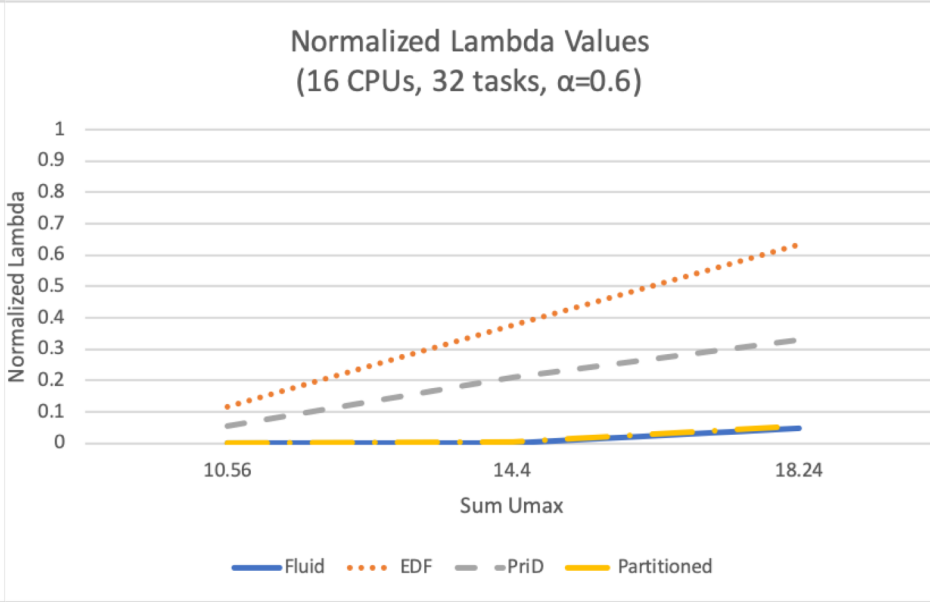

Higher is better



# Effects of changing no. of CPUs

(but keeping tasks/CPU constant)

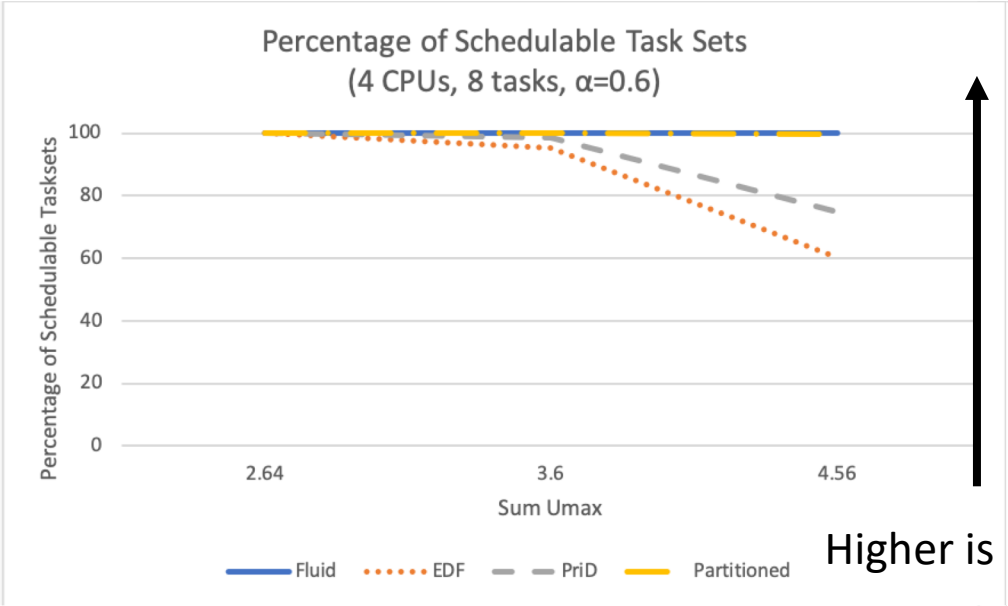



Lower is better

# Effects of changing no. of CPUs

(but keeping tasks/CPU constant)




Higher is better