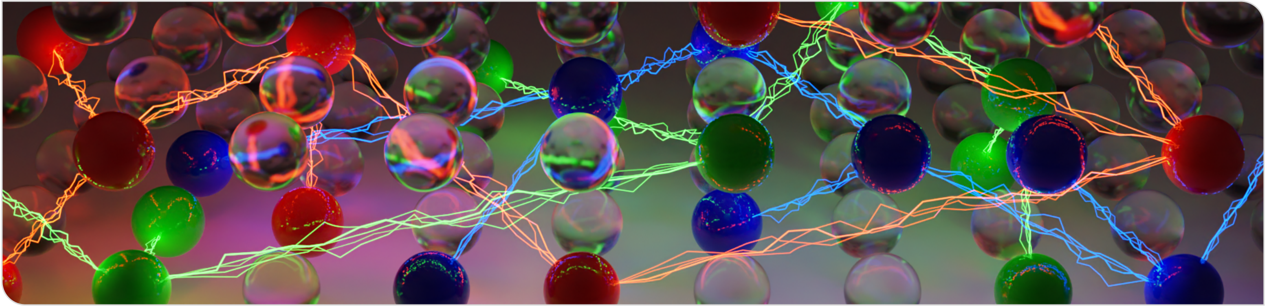
Karlsruhe Institute of Technology

European Research Council
Established by the European Commission

Decentralized Online Scheduling of Malleable NP-hard Jobs

28th International European Conference on Parallel and Distributed Computing (Euro-Par ’22)

Peter Sanders, Dominik Schreiber | August 24, 2022

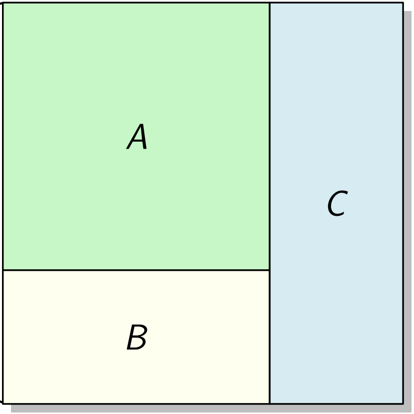


Motivation: Malleable Job Scheduling

https://doku.lrz.de/download/attachments/43320790/
image2019-11-15_12-48-5.png

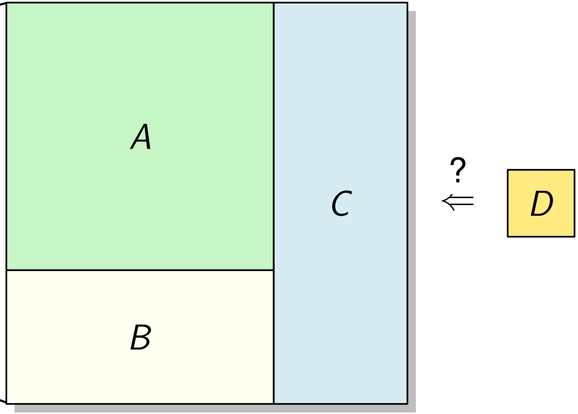
Motivation: Malleable Job Scheduling

https://doku.lrz.de/download/attachments/43320790/
image2019-11-15_12-48-5.png



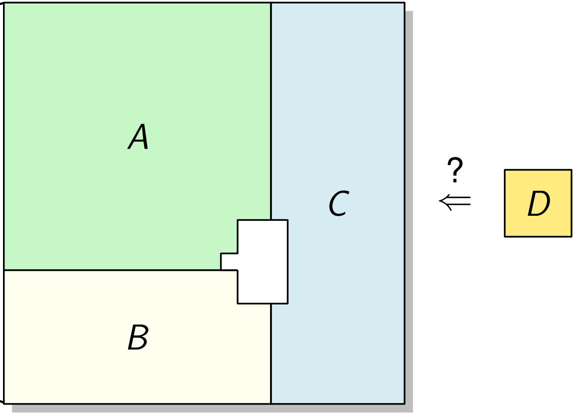
Motivation: Malleable Job Scheduling

https://doku.lrz.de/download/attachments/43320790/
image2019-11-15_12-48-5.png



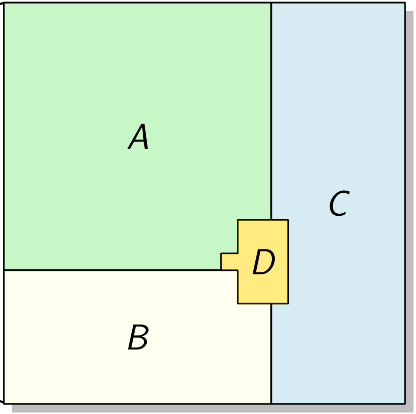
Motivation: Malleable Job Scheduling

https://doku.lrz.de/download/attachments/43320790/
image2019-11-15_12-48-5.png



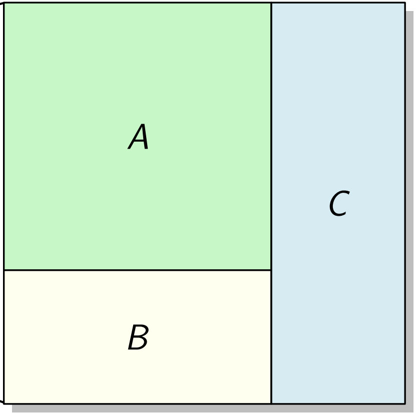
Motivation: Malleable Job Scheduling

https://doku.lrz.de/download/attachments/43320790/
image2019-11-15_12-48-5.png

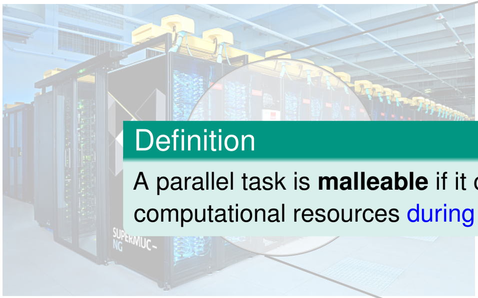


Motivation: Malleable Job Scheduling

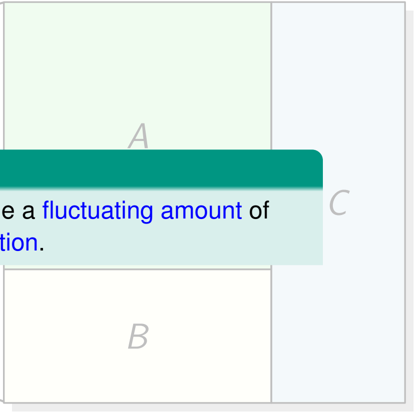
https://doku.lrz.de/download/attachments/43320790/
image2019-11-15_12-48-5.png



Motivation: Malleable Job Scheduling



https://doku.lrz.de/download/attachments/43320790/image2019-11-15_12-48-5.png



Definition

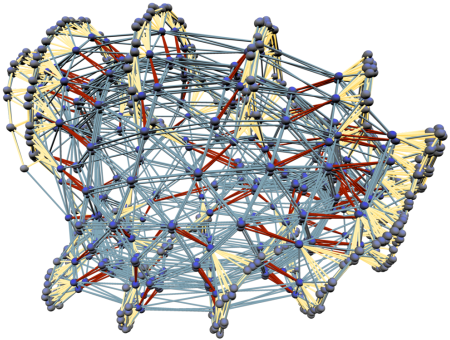
A parallel task is **malleable** if it can handle a fluctuating amount of computational resources during its execution.

Motivation: SAT Solving

Karlsruhe Institute of Technology

Propositional Satisfiability (SAT)

Input: Propositional formula F (Boolean variables combined with AND, OR, NOT)
Task: Find variable assignment s.t. F evaluates to `true`, or report that no such assignment exists



https://satcompetition.github.io/2022/logo2022-large.png

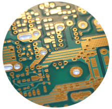
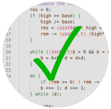
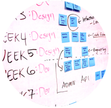
Motivation: SAT Solving

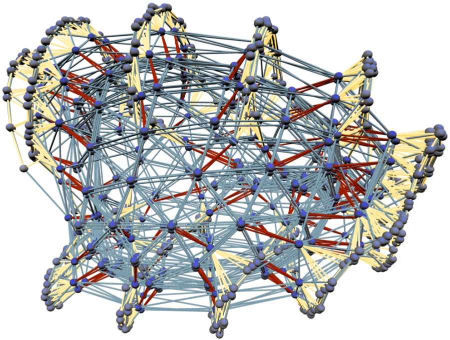
Karlsruhe Institute of Technology

Propositional Satisfiability (SAT)

Input: Propositional formula F (Boolean variables combined with AND, OR, NOT)
Task: Find variable assignment s.t. F evaluates to `true`, or report that no such assignment exists

- First problem proven **NP**-complete [Cook '71]
- Crucial buildling block for many applications

https://satcompetition.github.io/2022/logo2022-large.png

Malleability and SAT Solving: Why?

Karlsruhe Institute of Technology

Particular parallelization

- Portfolio of sequential solvers with diverse search strategies
- All solvers work on the entire problem, exchange knowledge periodically
- Limited scalability (despite significant advancements!)

Malleability and SAT Solving: Why?

Karlsruhe Institute of Technology

Particular parallelization

- Portfolio of sequential solvers with diverse search strategies
- All solvers work on the entire problem, exchange knowledge periodically
- Limited scalability (despite significant advancements!)

Unpredictability

- Processing time unknown in advance

Malleability and SAT Solving: Why?

Particular parallelization

- Portfolio of sequential solvers with diverse search strategies
- All solvers work on the entire problem, exchange knowledge periodically
- Limited scalability (despite significant advancements!)

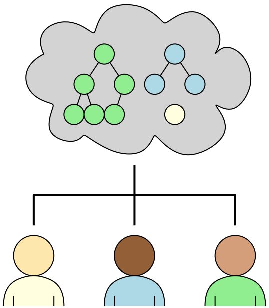
Unpredictability

- Processing time unknown in advance

Simple malleability

- Add or remove solvers to/from computation at will
- No global redistribution of data necessary
- Relatively small job descriptions

Vision and Contributions



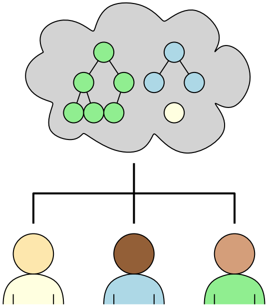
European Research Council
Established by the European Commission

On-Demand Service Platform for NP-hard Problems

Prior work: [Schreiber & Sanders, SAT'21]

- General system architecture
- Preliminary protocols for malleable scheduling
- Focus on award-winning SAT solving engine

Vision and Contributions



On-Demand Service Platform for NP-hard Problems

Prior work: [Schreiber & Sanders, SAT'21]

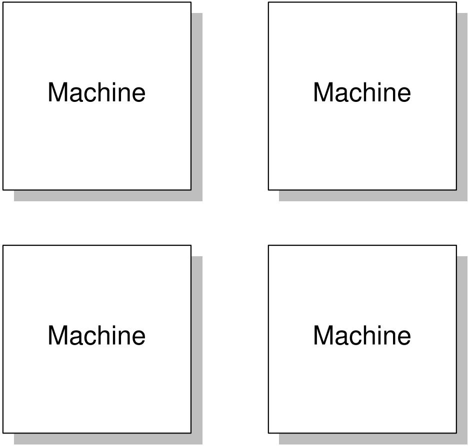
- General system architecture
- Preliminary protocols for malleable scheduling
- Focus on award-winning SAT solving engine

New contributions:

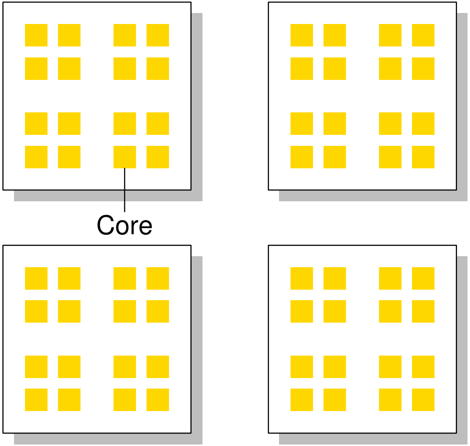
- Fully scalable decentralized scheduling algorithms
- Practical implementation for $\approx 10^4$ cores
- Extensive evaluation of scheduling performance, quality

European Research Council
Established by the European Commission

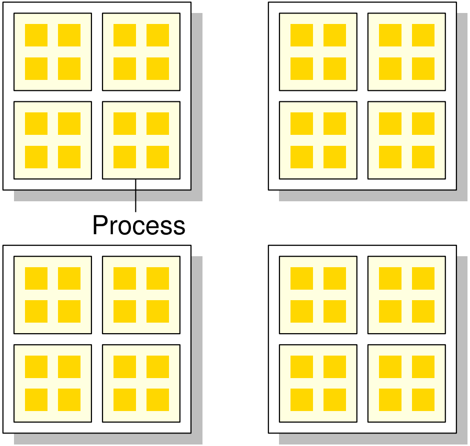
System Environment



System Environment

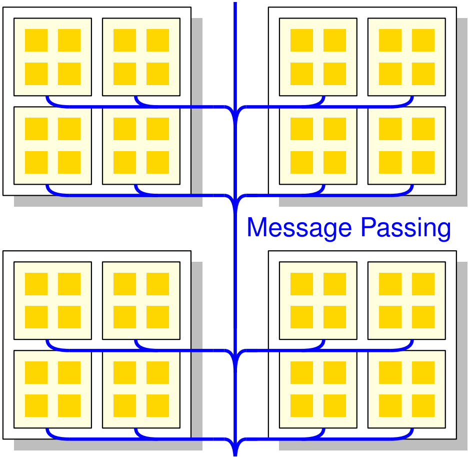


System Environment



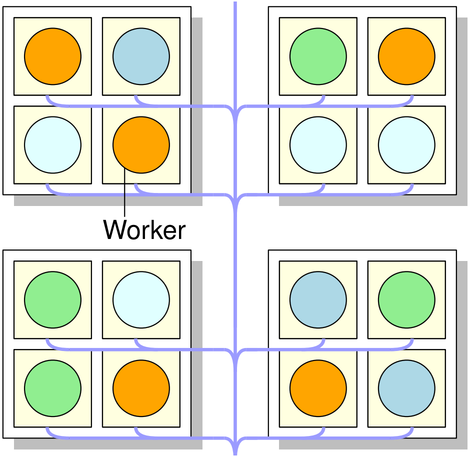
- m distributed processes

System Environment



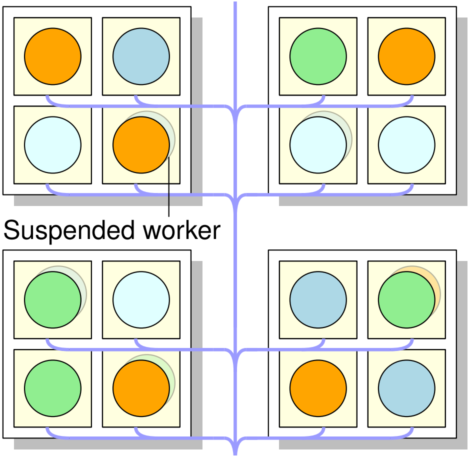
- *m* distributed processes

System Environment



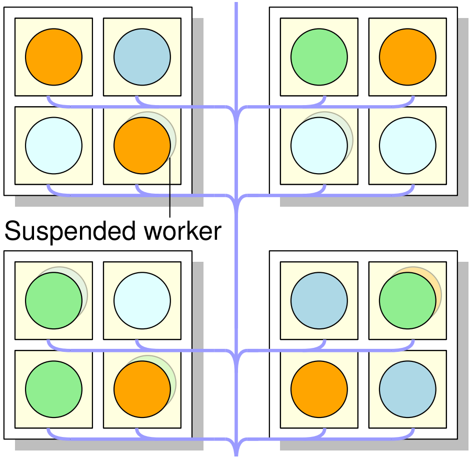
- *m* distributed processes
- Worker: Context of certain job on certain process

System Environment



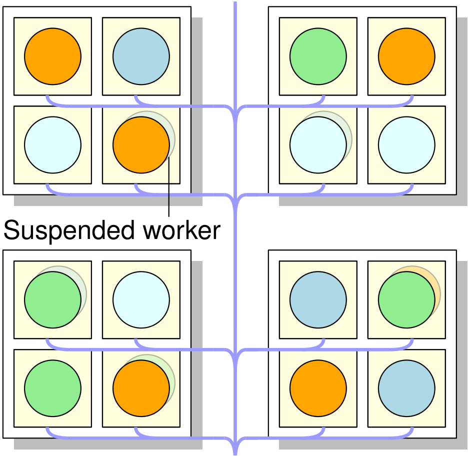
- m distributed processes
- Worker: Context of certain job on certain process
- Per process: ≤ 1 active worker ($\leq c$ suspended)

System Environment



- *m* distributed processes
- Worker: Context of certain job on certain process
- Per process: ≤ 1 active worker ($\leq c$ suspended)
- Processing and scheduling on the same cores!

System Environment



- m distributed processes
- Worker: Context of certain job on certain process
- Per process: ≤ 1 active worker ($\leq c$ suspended)
- Processing and scheduling on the same cores!

- Active jobs J, $n := |J|$
- Properties of each job j:
  - Priority $p_j \in \mathbb{R}^+$
  - Demand of resources $d_j \in \mathbb{N}^+$
  - Wallclock or CPU budget $b_j \in \mathbb{R}^+$ (optional)

Assigning Fair Volumes to Jobs

Karlsruhe Institute of Technology

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

Assigning Fair Volumes to Jobs

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \quad \vee \quad \sum_{j \in J} v_j = m$$

Assigning Fair Volumes to Jobs

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \quad \vee \quad \sum_{j \in J} v_j = m$$

(C2) Each job has at least one worker and at most d_j workers. $\forall j \in J : 1 \leq v_j \leq d_j$

Assigning Fair Volumes to Jobs

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \qquad \vee \qquad \sum_{j \in J} v_j = m$$

(C2) Each job has at least one worker and at most d_j workers. $\forall j \in J : 1 \leq v_j \leq d_j$

(C3) The volume of each job j scales proportionally with p_j as far as (C2) allows.

Assigning Fair Volumes to Jobs

Karlsruhe Institute of Technology

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \quad \vee \quad \sum_{j \in J} v_j = m$$

(C2) Each job has at least one worker and at most d_j workers. $\forall j \in J : 1 \leq v_j \leq d_j$

(C3) The volume of each job j scales proportionally with p_j as far as (C2) allows.

Theorem

Solving the above problem on m processes for $n \leq m$ jobs is possible in $\mathcal{O}(\log m)$ span.

Assigning Fair Volumes to Jobs

Karlsruhe Institute of Technology

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \quad \vee \quad \sum_{j \in J} v_j = m$$

(C2) Each job has at least one worker and at most d_j workers. $\forall j \in J : 1 \leq v_j \leq d_j$

(C3) The volume of each job j scales proportionally with p_j as far as (C2) allows.

Theorem

Solving the above problem on m processes for $n \leq m$ jobs is possible in $\mathcal{O}(\log m)$ span.

Central Idea

- Express job volumes $v_j = v_j(\alpha) := \alpha p_j$, $\alpha \geq 0$

Assigning Fair Volumes to Jobs

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \quad \vee \quad \sum_{j \in J} v_j = m$$

(C2) Each job has at least one worker and at most d_j workers. $\forall j \in J : 1 \leq v_j \leq d_j$

(C3) The volume of each job j scales proportionally with p_j as far as (C2) allows.

Theorem

Solving the above problem on m processes for $n \leq m$ jobs is possible in $\mathcal{O}(\log m)$ span.

Central Idea

- Express job volumes $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$

Assigning Fair Volumes to Jobs

Volume Assignment Problem

Map each job $j \in J$ to a fair number of workers, the volume $v_j \in \mathbb{N}^+$, s.t.

(C1) All job demands are fully met **or** all m processes are utilized.

$$(\forall j \in J : v_j = d_j) \quad \vee \quad \sum_{j \in J} v_j = m$$

(C2) Each job has at least one worker and at most d_j workers. $\forall j \in J : 1 \leq v_j \leq d_j$

(C3) The volume of each job j scales proportionally with p_j as far as (C2) allows.

Theorem

Solving the above problem on m processes for $n \leq m$ jobs is possible in $\mathcal{O}(\log m)$ span.

Central Idea

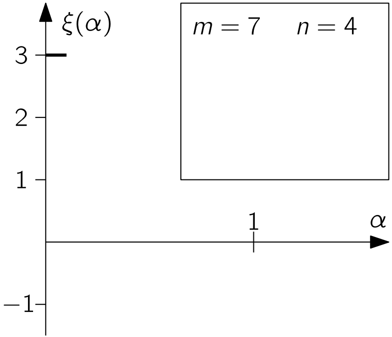
- Express job volumes $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Solve unused resources $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$ for $\xi(\alpha) = 0$

Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$

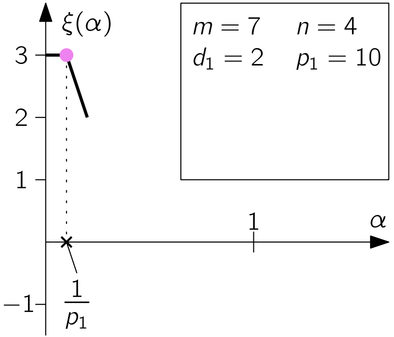
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$



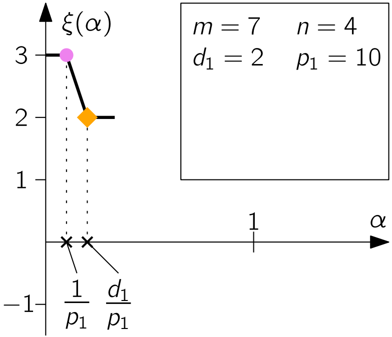
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$



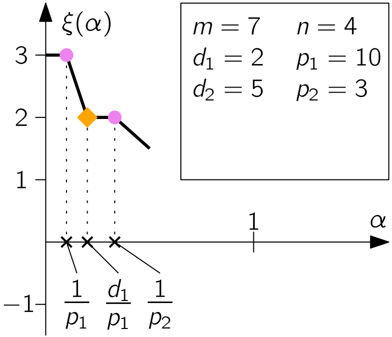
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$



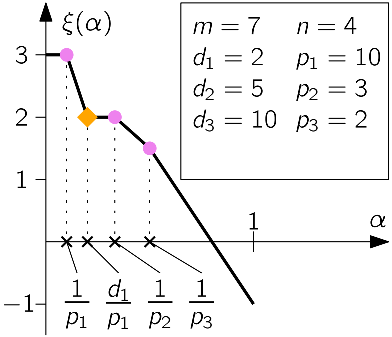
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$



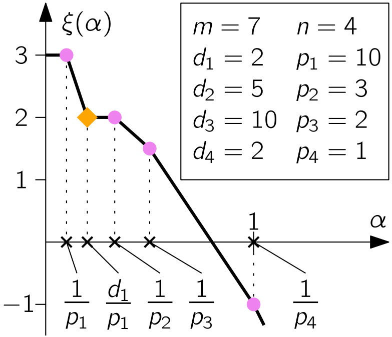
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$



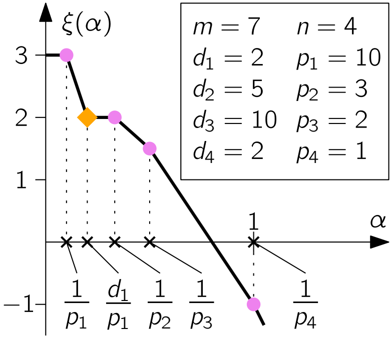
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) =$ $\max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$



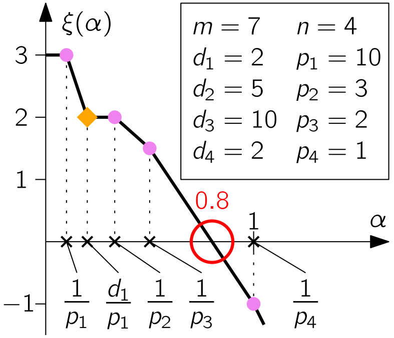
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes



Volume Assignment Algorithm

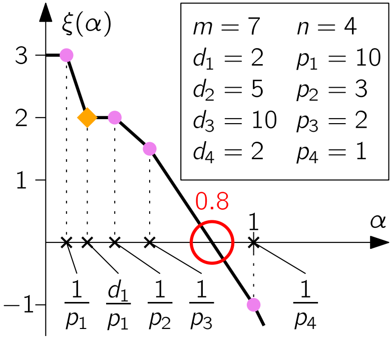
- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$



KIT Karlsruhe Institute of Technology

Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$



$v_1(\hat{\alpha}) = d_1 = 2$

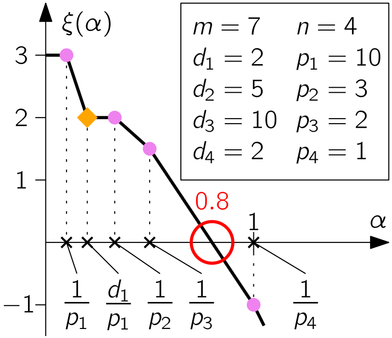
$v_2(\hat{\alpha}) = 0.8 \cdot 3 = 2.4$

$v_3(\hat{\alpha}) = 0.8 \cdot 2 = 1.6$

$v_4(\hat{\alpha}) = 1$

Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$
- Round the $v_j(\hat{\alpha})$ to appropriate integers



$v_1(\hat{\alpha}) = d_1 = 2$ $\quad v_1 = 2$

$v_2(\hat{\alpha}) = 0.8 \cdot 3 = 2.4$ $\quad v_2 = 3$

$v_3(\hat{\alpha}) = 0.8 \cdot 2 = 1.6$ $\quad v_3 = 1$

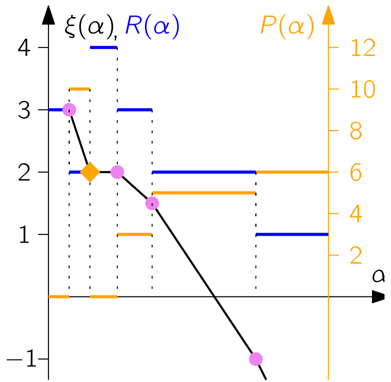
$v_4(\hat{\alpha}) = 1$ $\quad v_4 = 1$

Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$
- Round the $v_j(\hat{\alpha})$ to appropriate integers

*Parallel evaluation of ξ:**

- Compute ξ based on auxilliary terms R and P

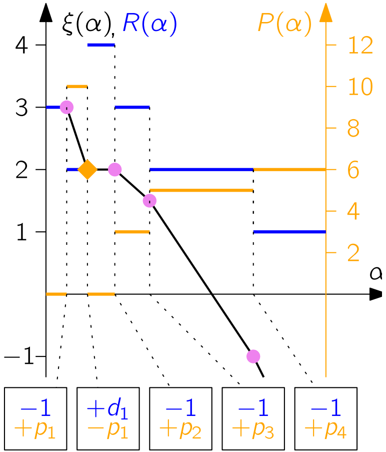


Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$
- Round the $v_j(\hat{\alpha})$ to appropriate integers

***Parallel evaluation of ξ:**

- Compute ξ based on auxilliary terms R and P
- Create, sort *events* which manipulate R and P

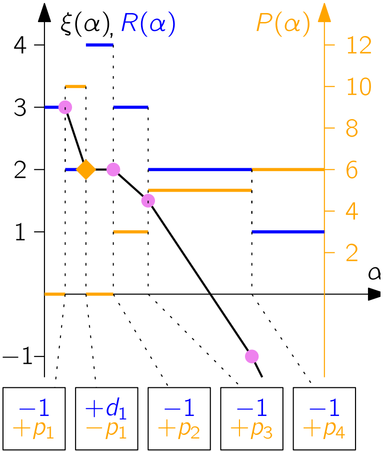


Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$
- Round the $v_j(\hat{\alpha})$ to appropriate integers

***Parallel evaluation of ξ:**

- Compute ξ based on auxilliary terms R and P
- Create, sort *events* which manipulate R and P
- Compute *all intermediate values* of R, P with prefix sum



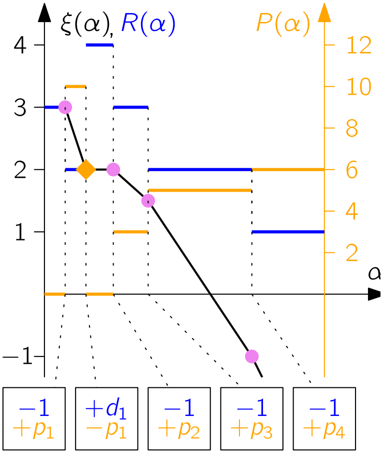
Volume Assignment Algorithm

- Job volumes $v_j = v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$, $\alpha \geq 0$
- Excess resources $\xi(\alpha) = m - \sum_{j \in J} v_j(\alpha)$
- Evaluate ξ in parallel* at all $2n$ values where ξ' changes
- Interpolate value $\hat{\alpha}$ where $\xi(\hat{\alpha}) = 0$
- Round the $v_j(\hat{\alpha})$ to appropriate integers

***Parallel evaluation of ξ:**

- Compute ξ based on auxilliary terms R and P
- Create, sort *events* which manipulate R and P
- Compute *all intermediate values* of R, P with prefix sum

All-reductions, prefix sums, sorting $\mathcal{O}(m)$ elements:
Possible in $\mathcal{O}(\log m)$ time [Ajtai, Komlós, Szemerédi '83]

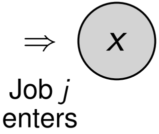


From Volumes to Scheduling

Each job j receives v_j processes. **Which processes?**

From Volumes to Scheduling

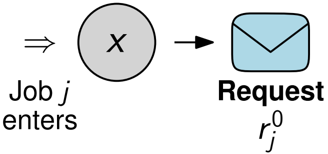
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

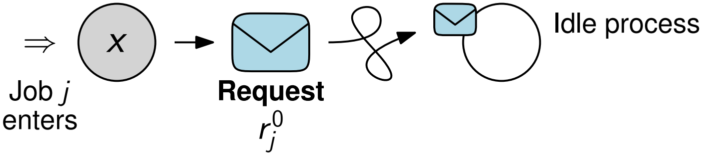
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

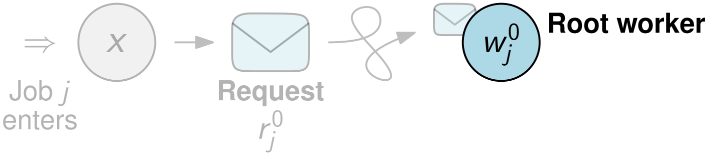
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

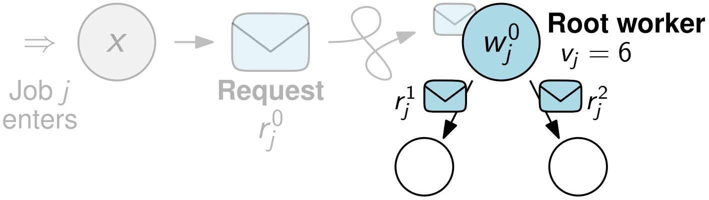
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

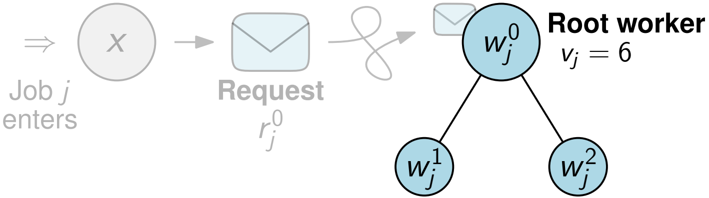
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

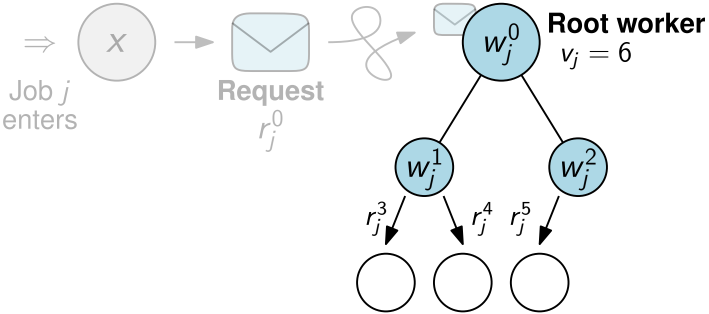
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

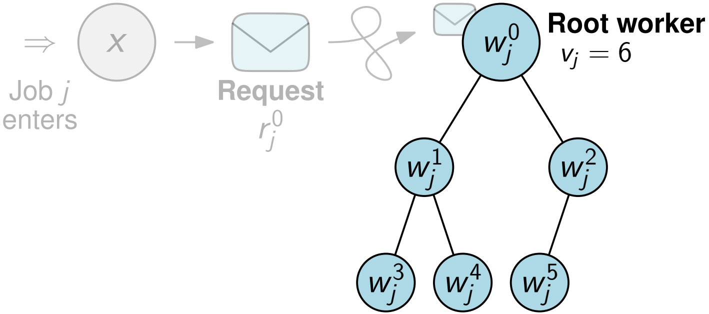
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

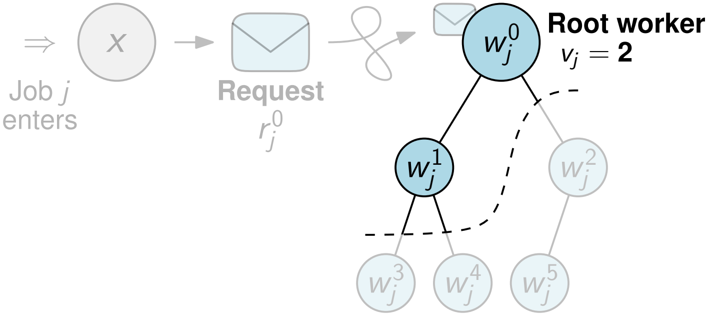
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

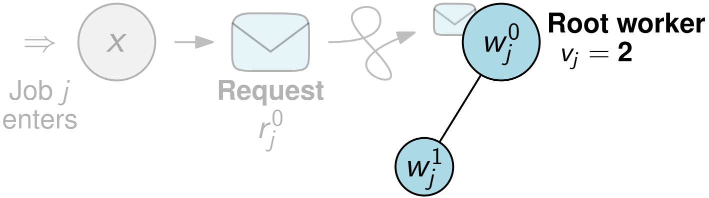
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

From Volumes to Scheduling

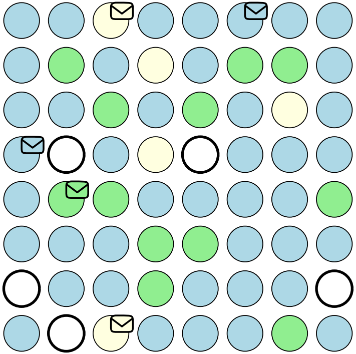
Each job j receives v_j processes. **Which processes?**



[Schreiber & Sanders, SAT'21]

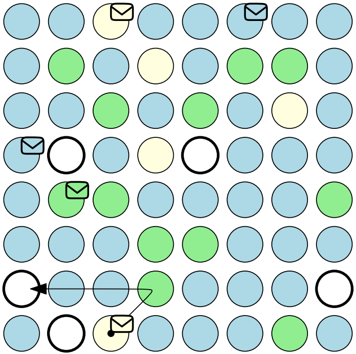
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



Matching Requests and Idle Processes

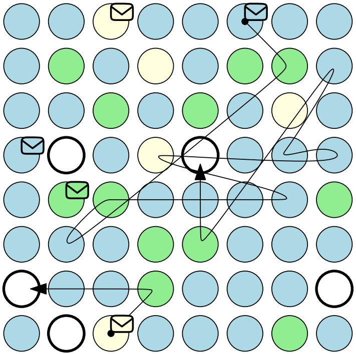
Random walks [Schreiber & Sanders, SAT'21]



Matching Requests and Idle Processes

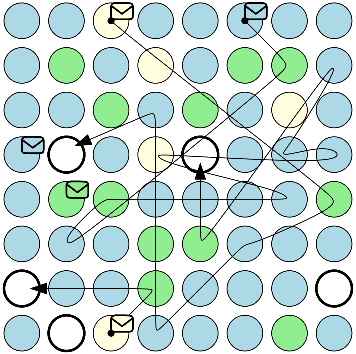
Karlsruhe Institute of Technology

Random walks [Schreiber & Sanders, SAT'21]



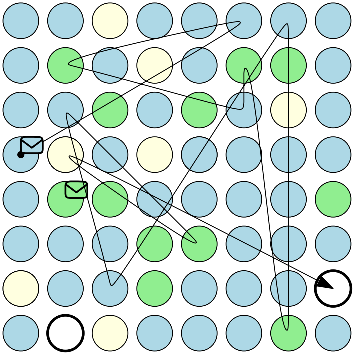
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



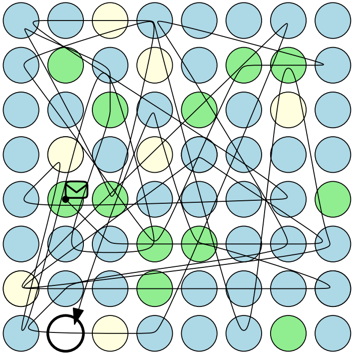
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



Matching Requests and Idle Processes

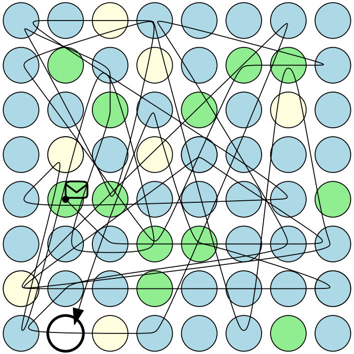
Random walks [Schreiber & Sanders, SAT'21]



Matching Requests and Idle Processes

Karlsruhe Institute of Technology

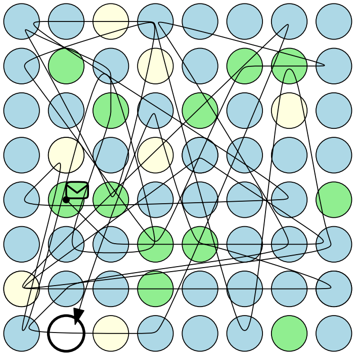
Random walks [Schreiber & Sanders, SAT'21]



⇒ Deliberately leave some processes idle
OR risk high scheduling latencies

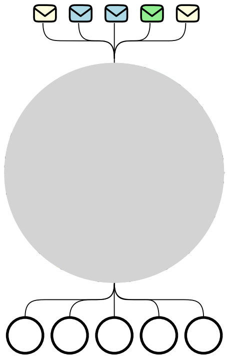
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



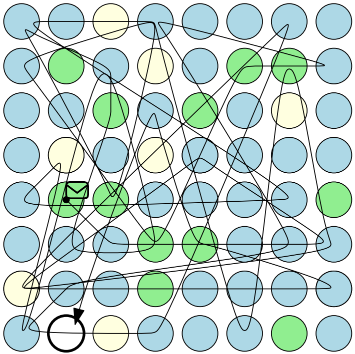
⇒ Deliberately leave some processes idle
OR risk high scheduling latencies

Collective Matching



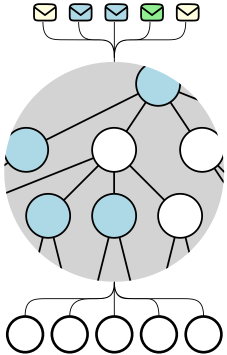
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



$\Rightarrow$ Deliberately leave some processes idle
OR risk high scheduling latencies

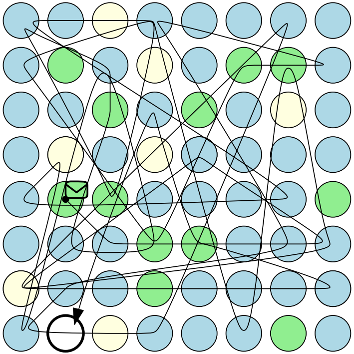
Collective Matching



Implemented: Route requests along tree

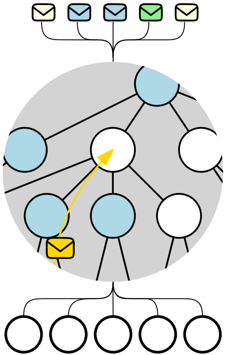
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



⇒ Deliberately leave some processes idle
OR risk high scheduling latencies

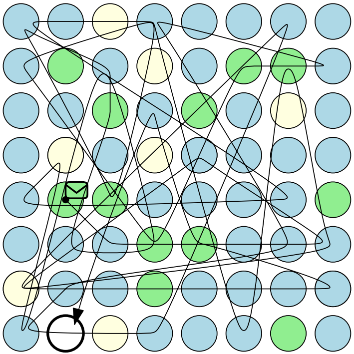
Collective Matching



Implemented: Route requests along tree

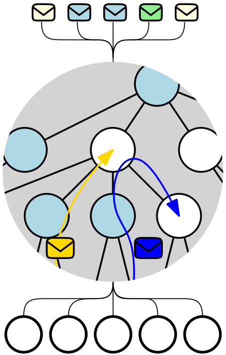
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



⇒ Deliberately leave some processes idle
OR risk high scheduling latencies

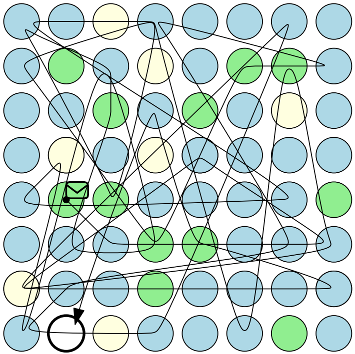
Collective Matching



Implemented: Route requests along tree

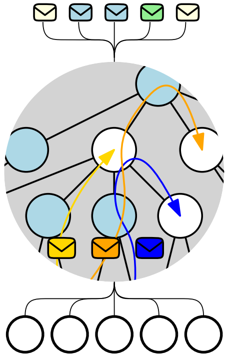
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



⇒ Deliberately leave some processes idle
OR risk high scheduling latencies

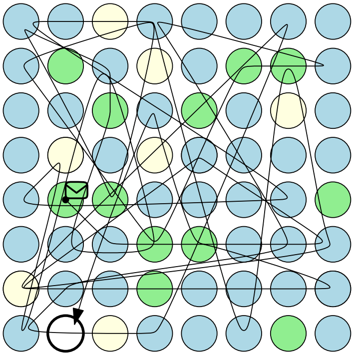
Collective Matching



Implemented: Route requests along tree

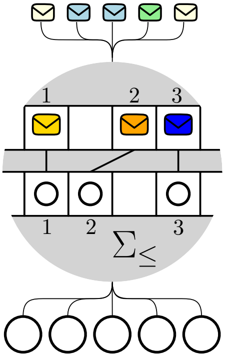
Matching Requests and Idle Processes

Random walks [Schreiber & Sanders, SAT'21]



$\Rightarrow$ Deliberately leave some processes idle
OR risk high scheduling latencies

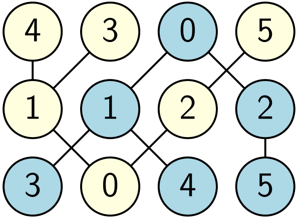
Collective Matching



Implemented: Route requests along tree
Theory: $\mathcal{O}(\log m)$ span via prefix sums

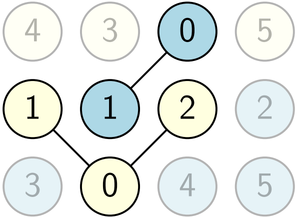
Reusing Suspended Workers

Naïve scheduling



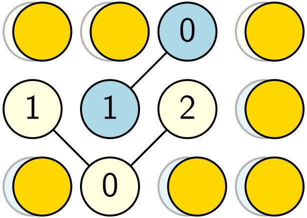
Reusing Suspended Workers

Naïve scheduling



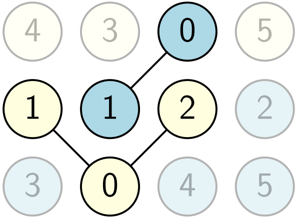
Reusing Suspended Workers

Naïve scheduling



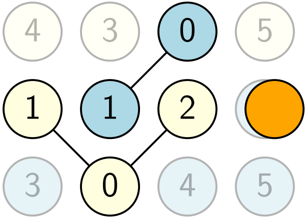
Reusing Suspended Workers

Naïve scheduling



Reusing Suspended Workers

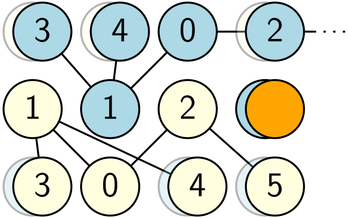
Naïve scheduling



- Idle procs. can be seized by other jobs

Reusing Suspended Workers

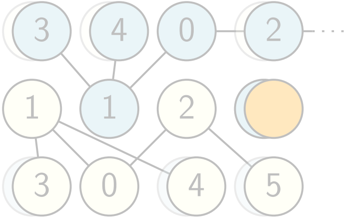
Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

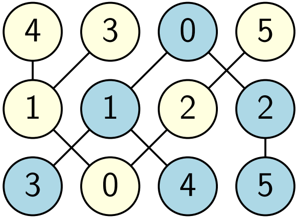
Reusing Suspended Workers

Naïve scheduling



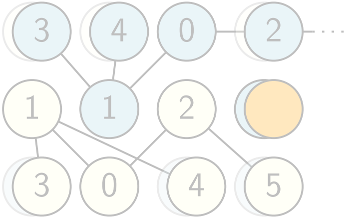
- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

Improvements



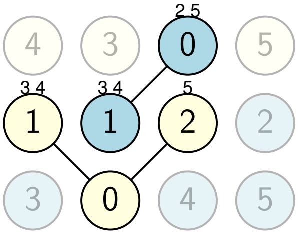
Reusing Suspended Workers

Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

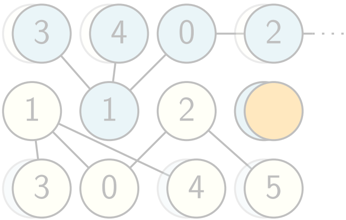
Improvements



- Workers transitively remember past children

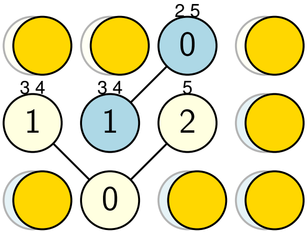
Reusing Suspended Workers

Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

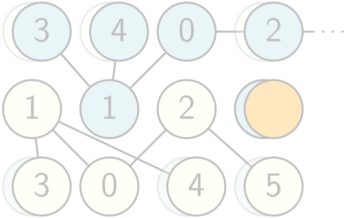
Improvements



- Workers transitively remember past children

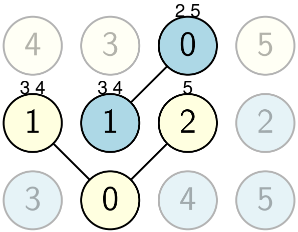
Reusing Suspended Workers

Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

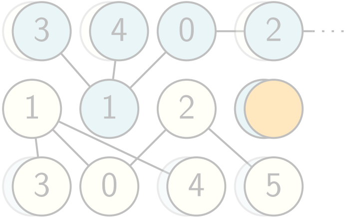
Improvements



- Workers transitively remember past children

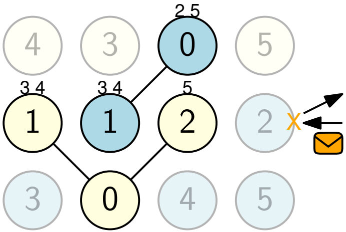
Reusing Suspended Workers

Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

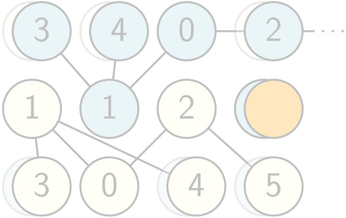
Improvements



- Workers transitively remember past children
- Idle processes prefer to reactivate past child

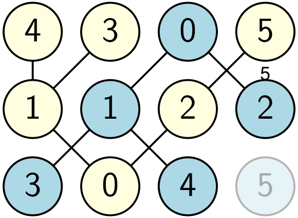
Reusing Suspended Workers

Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

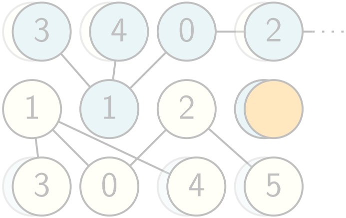
Improvements



- Workers transitively remember past children
- Idle processes prefer to reactivate past child

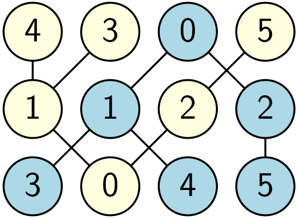
Reusing Suspended Workers

Naïve scheduling



- Idle procs. can be seized by other jobs
- Jobs can re-grow differently after shrinking

Improvements



- Workers transitively remember past children
- Idle processes prefer to reactivate past child

Evaluation: Context

The story thus far: **Malleable SAT solving is effective & efficient** [Schreiber & Sanders, SAT'21]

- Decent SAT solving performance if job volumes fluctuate
- Lower response times with malleability than solving each formula at fixed, small scale
- Better resource efficiency than sequentially scheduling massively parallel SAT solver

Evaluation: Context

The story thus far: **Malleable SAT solving is effective & efficient** [Schreiber & Sanders, SAT'21]

- Decent SAT solving performance if job volumes fluctuate
- Lower response times with malleability than solving each formula at fixed, small scale
- Better resource efficiency than sequentially scheduling massively parallel SAT solver

Now: **Performance and quality of our scheduling**

Evaluation: Setup

Environment

SuperMUC-NG (#26 @ TOP500 '22)
$\leq$ 128 nodes $\times$ 48 cores @ 2.7 GHz
SuSE Linux Enterprise Server

Evaluation: Setup

Karlsruhe Institute of Technology

Environment

SuperMUC-NG (#26 @ TOP500 ’22)
$\leq$ 128 nodes $\times$ 48 cores @ 2.7 GHz
SuSE Linux Enterprise Server

Implementation

C++17 with MPI
+ multi-threading;
simplified volume calc.

Evaluation: Setup

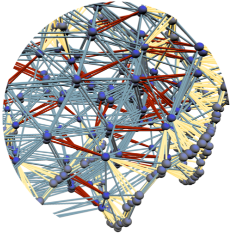
Environment

SuperMUC-NG (#26 @ TOP500 '22)
$\leq$ 128 nodes $\times$ 48 cores @ 2.7 GHz
SuSE Linux Enterprise Server

Implementation

C++17 with MPI
+ multi-threading;
simplified volume calc.

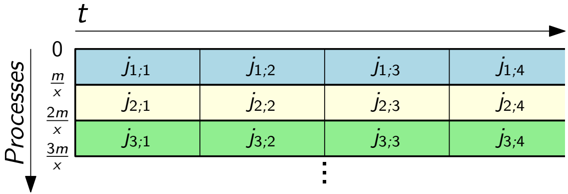
Inputs



400 instances from
Int. SAT Competition 2020

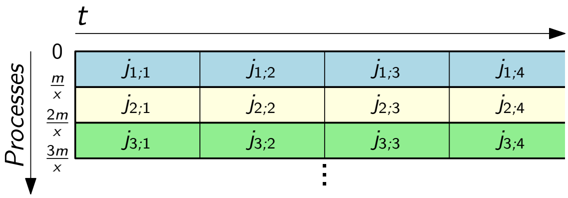
Efficiency for Uniform Jobs

Setup: 1536 processes × four cores
Scenario: x jobs in parallel with fixed CPU limit



Efficiency for Uniform Jobs

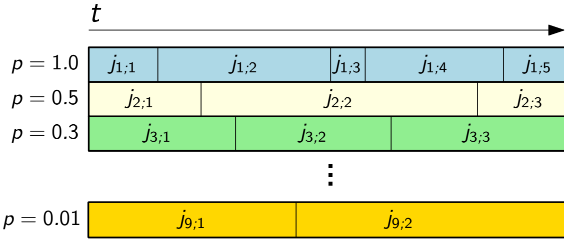
Setup: 1536 processes × four cores
Scenario: x jobs in parallel with fixed CPU limit



Efficiency: >96% for reasonable loads

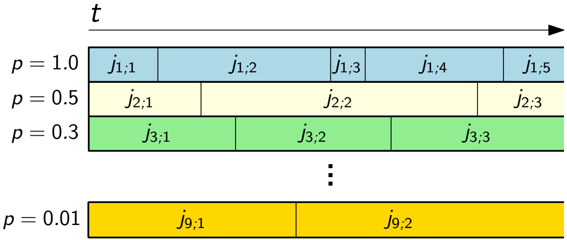
Impact of Job Priorities

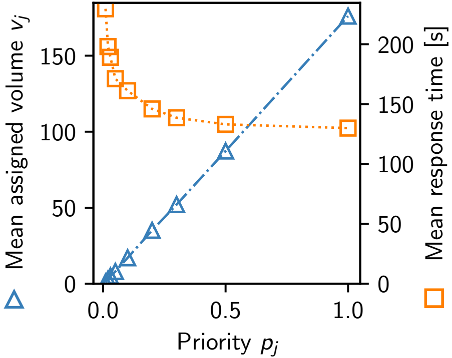
Setup: 384 processes × four cores
Scenario: 9 “clients” introducing jobs sequentially



Impact of Job Priorities

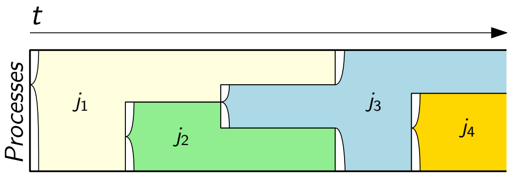
Setup: 384 processes × four cores
Scenario: 9 "clients" introducing jobs sequentially





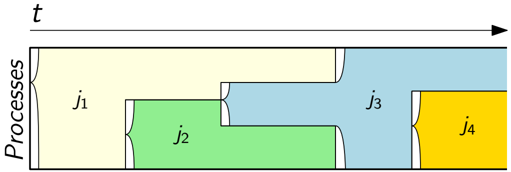
Realistic Arrival Rates: Overview

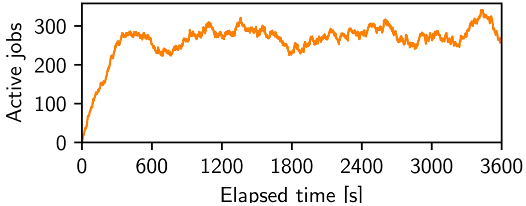
Setup: 1536 processes $\times$ four cores
Scenario: random arrival of ISC20 jobs
with random demands, priorities, time limits

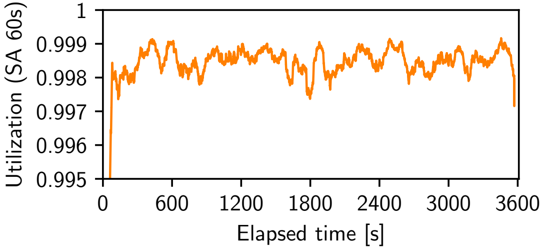


Realistic Arrival Rates: Overview

Setup: 1536 processes × four cores
Scenario: random arrival of ISC20 jobs
with random demands, priorities, time limits



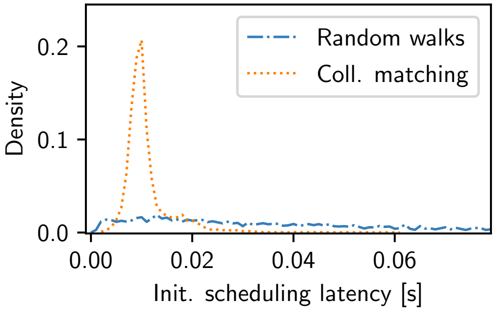




Realistic Arrival Rates: Scheduling Quality

Request matching strategies:

- Random walks [Schreiber & Sanders, SAT'21]
- Collective matching along tree of processes



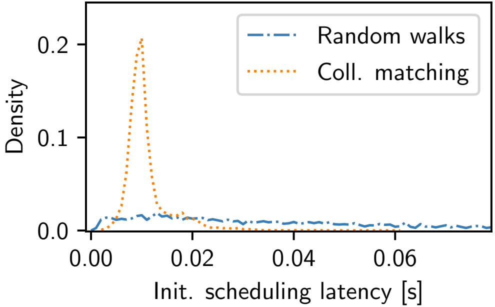
Realistic Arrival Rates: Scheduling Quality

Request matching strategies:

- Random walks [Schreiber & Sanders, SAT'21]
- Collective matching along tree of processes

Average Latencies

$\approx$ 10 ms for scheduling 1st worker
$\approx$ 1 ms for calculating fair volumes
$\approx$ 6 ms for doubling a job's size



Realistic Arrival Rates: Scheduling Quality

Request matching strategies:

- Random walks [Schreiber & Sanders, SAT'21]
- Collective matching along tree of processes

Average Latencies

$\approx$ 10 ms for scheduling 1st worker
$\approx$ 1 ms for calculating fair volumes
$\approx$ 6 ms for doubling a job's size

Stability of Scheduling

How many more workers are created than strictly necessary for each job?

Realistic Arrival Rates: Scheduling Quality

Request matching strategies:

- Random walks [Schreiber & Sanders, SAT'21]
- Collective matching along tree of processes

Average Latencies

$\approx$ 10 ms for scheduling 1st worker
$\approx$ 1 ms for calculating fair volumes
$\approx$ 6 ms for doubling a job's size

Stability of Scheduling

How many more workers are created than strictly necessary for each job?

Median: 40% $\Rightarrow$ 25%

Realistic Arrival Rates: Scheduling Quality

Request matching strategies:

- Random walks [Schreiber & Sanders, SAT'21]
- Collective matching along tree of processes

Average Latencies

$\approx$ 10 ms for scheduling 1st worker
$\approx$ 1 ms for calculating fair volumes
$\approx$ 6 ms for doubling a job's size

Stability of Scheduling

How many more workers are created than strictly necessary for each job?

Median: 40% $\Rightarrow$ 25%

How probable is it that any given worker w_j^i has been initialized only once?

Realistic Arrival Rates: Scheduling Quality

Request matching strategies:

- Random walks [Schreiber & Sanders, SAT'21]
- Collective matching along tree of processes

Average Latencies

$\approx$ 10 ms for scheduling 1st worker
$\approx$ 1 ms for calculating fair volumes
$\approx$ 6 ms for doubling a job's size

Stability of Scheduling

How many more workers are created than strictly necessary for each job?

Median: 40% $\Rightarrow$ 25%

How probable is it that any given worker w_j^i has been initialized only once?

89%

Conclusion

Karlsruhe Institute of Technology

Recap

- Decentralized online scheduling of malleable jobs with priorities, demands, unknown processing time
- Theory: Scalable algorithms with logarithmic span
- Experiments (6144 c.): Scheduling delays in range of milliseconds, near-optimal utilization

Future work

- Integration of further applications (*k*-Means clustering, hierarchical planning)
- Fault tolerance, heterogeneous systems
- Better handling of fractional resources

Many thanks to all reviewers!

Certified Euro-Par Artifact

Published at Journal of Open Source Software

Best large-scale SAT solver since 2020

Ready for cloud deployment

docker

aws

Software archived

zenodo

figshare

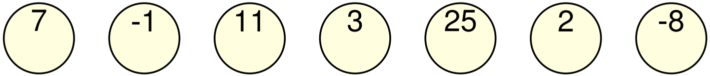
Developed at

/domschrei /mallob

Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

- **All-reduction** – e.g., broadcast value, compute maximum



Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

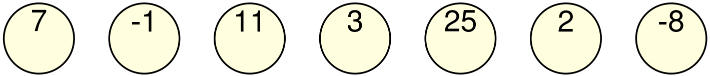
- **All-reduction** – e.g., broadcast value, compute maximum

Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

- **All-reduction** – e.g., broadcast value, compute maximum

- **Sorting** $\mathcal{O}(m)$ scattered elements [Ajtai, Komlós, Szemerédi '83]

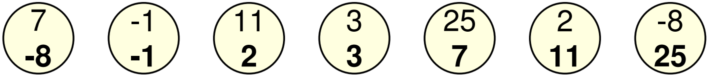


Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

- **All-reduction** – e.g., broadcast value, compute maximum

- **Sorting** $\mathcal{O}(m)$ scattered elements [Ajtai, Komlós, Szemerédi ’83]



Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

- **All-reduction** – e.g., broadcast value, compute maximum

7	-1	11	3	25	2	-8
25	**25**	**25**	**25**	**25**	**25**	**25**

- **Sorting** $\mathcal{O}(m)$ scattered elements [Ajtai, Komlós, Szemerédi ’83]

7	-1	11	3	25	2	-8
-8	**-1**	**2**	**3**	**7**	**11**	**25**

- **Prefix sum** (or “Scan”)

7	-1	11	3	25	2	-8

Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

- **All-reduction** – e.g., broadcast value, compute maximum

7	-1	11	3	25	2	-8
25	**25**	**25**	**25**	**25**	**25**	**25**

- **Sorting** $\mathcal{O}(m)$ scattered elements [Ajtai, Komlós, Szemerédi ’83]

7	-1	11	3	25	2	-8
-8	**-1**	**2**	**3**	**7**	**11**	**25**

- **Prefix sum** (or “Scan”)

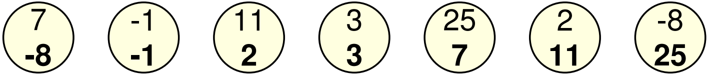
7	-1	11	3	25	2	-8
7	**6**	**17**	**20**	**45**	**47**	**39**

Volume Assignment: Prerequisites

Collective operations with $\mathcal{O}(\log m)$ *span* / *depth*:

- **All-reduction** – e.g., broadcast value, compute maximum

- **Sorting** $\mathcal{O}(m)$ scattered elements [Ajtai, Komlós, Szemerédi ’83]



- **Prefix sum** (or “Scan”)

Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) :=$ αp_j for $\alpha \in \mathbb{R}_0^+$

Volume Assignment (1/2)

Karlsruhe Institute of Technology

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

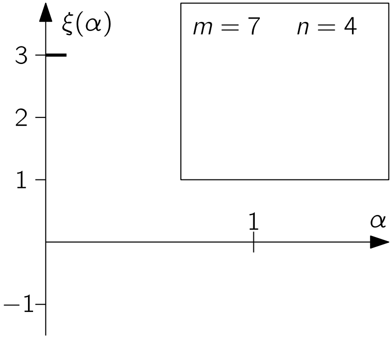
- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$

Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

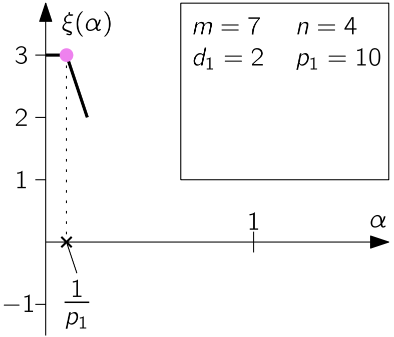
- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$



Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

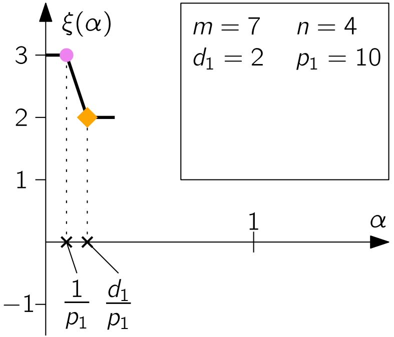


- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$

Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process



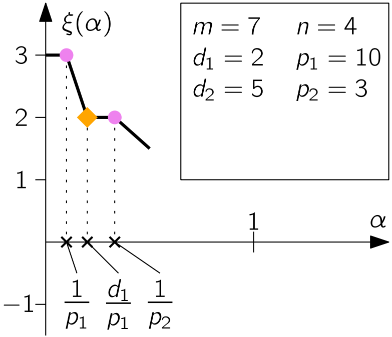
- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$

Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$

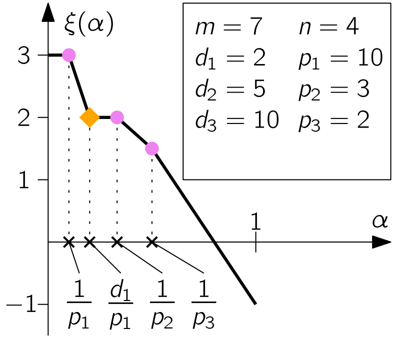


Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$

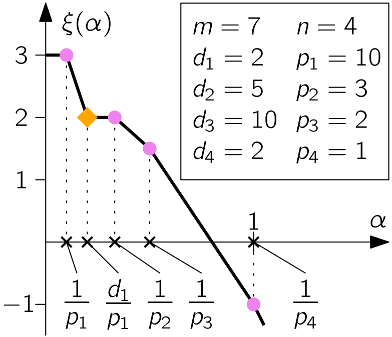


Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$

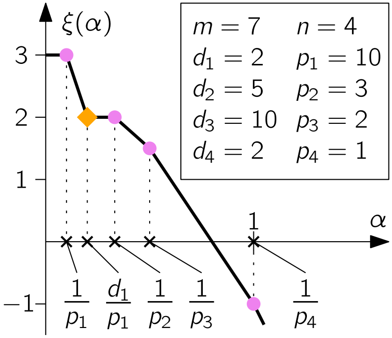


Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$
- **Evaluate** ξ at all $2n$ values where its gradient changes

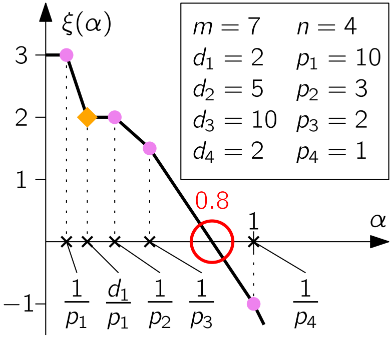


Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$
- **Evaluate** ξ at all $2n$ values where its gradient changes
- **Interpolate** value $\hat{\alpha}$ where ξ changes its sign

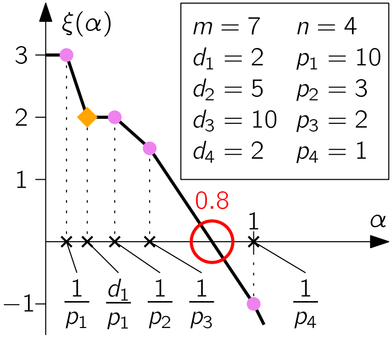


Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$
- **Evaluate** ξ at all $2n$ values where its gradient changes
- **Interpolate** value $\hat{\alpha}$ where ξ changes its sign



$m = 7$	$n = 4$
$d_1 = 2$	$p_1 = 10$
$d_2 = 5$	$p_2 = 3$
$d_3 = 10$	$p_3 = 2$
$d_4 = 2$	$p_4 = 1$

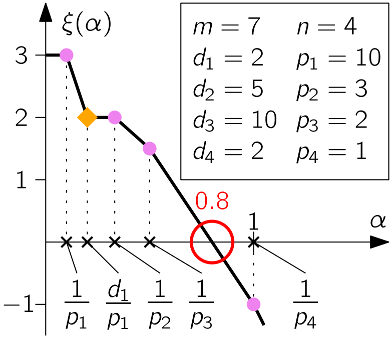
$v_1(\hat{\alpha}) = d_1 = 2$
$v_2(\hat{\alpha}) = 0.8 \cdot 3 = 2.4$
$v_3(\hat{\alpha}) = 0.8 \cdot 2 = 1.6$
$v_4(\hat{\alpha}) = 1$

Volume Assignment (1/2)

Assumptions

- m processes, $n \leq m$ jobs
- Each job is represented by one particular process

- Parametrize $v_j = v_j(\alpha) := \max(1, \min(d_j, \alpha p_j))$ for $\alpha \in \mathbb{R}_0^+$
- Express *unused resources* as $\xi(\alpha) := m - \sum_{j \in J} v_j(\alpha)$
- **Evaluate** ξ at all $2n$ values where its gradient changes
- **Interpolate** value $\hat{\alpha}$ where ξ changes its sign
- **Round** the $v_j(\hat{\alpha})$ to appropriate integers



$m = 7$	$n = 4$
$d_1 = 2$	$p_1 = 10$
$d_2 = 5$	$p_2 = 3$
$d_3 = 10$	$p_3 = 2$
$d_4 = 2$	$p_4 = 1$

$v_1(\hat{\alpha}) = d_1 = 2$ $\quad v_1 = 2$

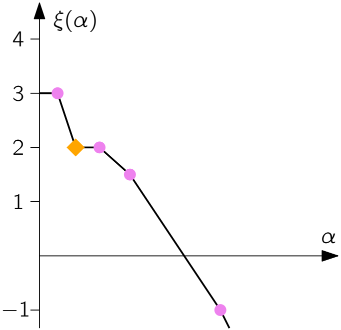
$v_2(\hat{\alpha}) = 0.8 \cdot 3 = 2.4$ $\quad v_2 = 3$

$v_3(\hat{\alpha}) = 0.8 \cdot 2 = 1.6$ $\quad v_3 = 1$

$v_4(\hat{\alpha}) = 1$ $\quad v_4 = 1$

Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

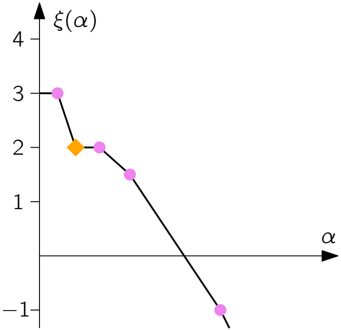


Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

- Alternative formulation: $\xi(\alpha) = m - R - \alpha P$

$$R = \sum_{j:\alpha p_j < 1} 1 + \sum_{j:\alpha p_j > d_j} d_j \qquad P = \sum_{j:1 \le \alpha p_j \le d_j} p_j$$

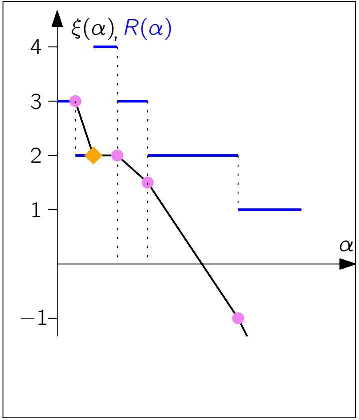


Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

- Alternative formulation: $\xi(\alpha) \;=\; m - R - \alpha\, P$

$$R = \sum_{j\,:\alpha p_j<1} 1 \;+ \sum_{j\,:\alpha p_j>d_j} d_j \qquad P = \sum_{j\,:1\le\alpha p_j\le d_j} p_j$$

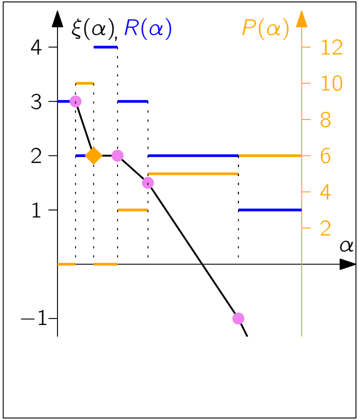


Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

- Alternative formulation: $\xi(\alpha) = m - R - \alpha P$

$$R = \sum_{j:\alpha p_j < 1} 1 + \sum_{j:\alpha p_j > d_j} d_j \qquad P = \sum_{j:1 \le \alpha p_j \le d_j} p_j$$



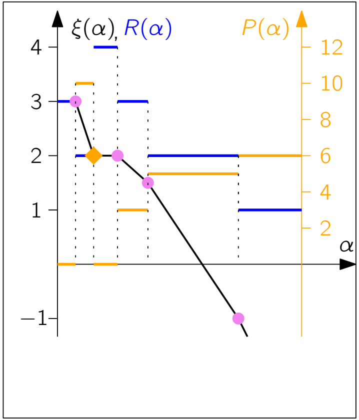
Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

- Alternative formulation: $\xi(\alpha) = m - R - \alpha P$

  $R = \sum_{j:\alpha p_j<1} 1 + \sum_{j:\alpha p_j>d_j} d_j \qquad P = \sum_{j:1\leq\alpha p_j\leq d_j} p_j$

- Express $2n$ points as **events** manipulating R, P



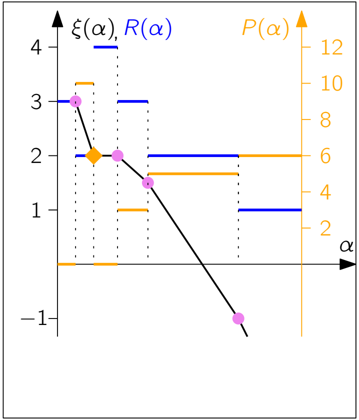
Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

- Alternative formulation: $\xi(\alpha) = m - R - \alpha P$

  $$R = \sum_{j:\alpha p_j < 1} 1 + \sum_{j:\alpha p_j > d_j} d_j \qquad P = \sum_{j:1 \le \alpha p_j \le d_j} p_j$$

- Express $2n$ points as **events** manipulating R, P
- **Sort** events in parallel
- Compute **prefix sum** $(R_{\le e}, P_{\le e})$ over sorted events e



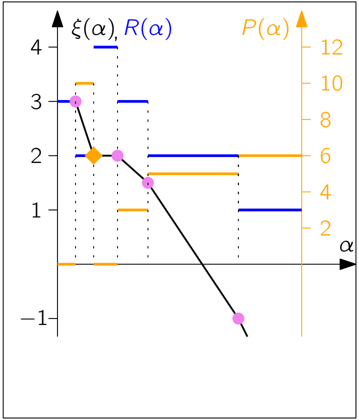
Volume Assignment (2/2)

Evaluate excess function ξ at all $2n$ points of interest **in parallel**:

- Alternative formulation: $\xi(\alpha) = m - R - \alpha P$

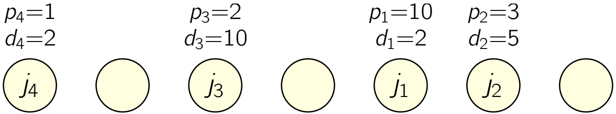
  $$R = \sum_{j:\alpha p_j < 1} 1 + \sum_{j:\alpha p_j > d_j} d_j \qquad P = \sum_{j:1 \leq \alpha p_j \leq d_j} p_j$$

- Express $2n$ points as **events** manipulating R, P
- **Sort** events in parallel
- Compute **prefix sum** $(R_{\leq e}, P_{\leq e})$ over sorted events e
- **Evaluate** $\xi(e) = m - (n + R_{\leq e}) - e P_{\leq e}$



Volume Assignment: Algorithm

$m = 7$



$$v_j(\alpha) \;=\; \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha)$$

Volume Assignment: Algorithm

$m = 7$

	$p_4=1$, $d_4=2$		$p_3=2$, $d_3=10$		$p_1=10$, $d_1=2$	$p_2=3$, $d_2=5$	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0

$$v_j(\alpha) \;=\; \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha)$$

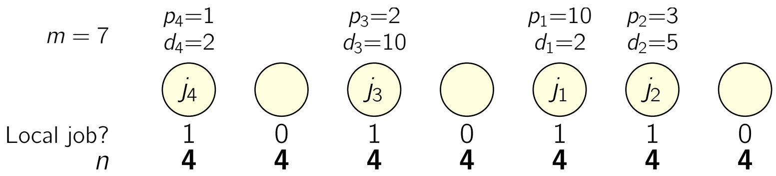
Volume Assignment: Algorithm

$m = 7$							
	p_4=1, d_4=2		p_3=2, d_3=10		p_1=10, d_1=2	p_2=3, d_2=5	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha)$$

Volume Assignment: Algorithm



$$v_j(\alpha) \;=\; \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \textstyle\sum_j v_j(\alpha)$$
$$= m - R - \alpha \; P$$

$$R = \textstyle\sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$
$$P = \textstyle\sum_{j:1<\alpha p_j < d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$	$p_4{=}1$, $d_4{=}2$		$p_3{=}2$, $d_3{=}10$		$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**
Events $\alpha{=}1/p_j$	$\alpha{=}1$: $R{-}{=}1$ $P{+}{=}1$		$\alpha{=}0.5$: $R{-}{=}1$ $P{+}{=}2$		$\alpha{=}0.1$: $R{-}{=}1$ $P{+}{=}10$	$\alpha{=}1/3$: $R{-}{=}1$ $P{+}{=}3$	

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$	$p_4{=}1$, $d_4{=}2$		$p_3{=}2$, $d_3{=}10$		$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**
Events $\alpha{=}1/p_j$	$\alpha{=}1$: $R{-}{=}1$ $P{+}{=}1$		$\alpha{=}0.5$: $R{-}{=}1$ $P{+}{=}2$		$\alpha{=}0.1$: $R{-}{=}1$ $P{+}{=}10$	$\alpha{=}1/3$: $R{-}{=}1$ $P{+}{=}3$	
Events $\alpha{=}d_j/p_j$	$\alpha{=}2$: $R{+}{=}2$ $P{-}{=}1$		$\alpha{=}5$: $R{+}{=}10$ $P{-}{=}2$		$\alpha{=}0.2$: $R{+}{=}2$ $P{-}{=}10$	$\alpha{=}5/3$: $R{+}{=}5$ $P{-}{=}3$	

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \le 1} 1 + \sum_{j:\alpha p_j \ge d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$

	p_4=1, d_4=2		p_3=2, d_3=10		p_1=10, d_1=2	p_2=3, d_2=5	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Sorted events							
α=0.1:	α=0.2:	α=1/3:	α=0.5:	α=1:	α=5/3:	α=2:	α=5:
$R-$=1	$R+$=2	$R-$=1	$R-$=1	$R-$=1	$R+$=5	$R+$=2	$R+$=10
$P+$=10	$P-$=10	$P+$=3	$P+$=2	$P+$=1	$P-$=3	$P-$=1	$P-$=2

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$

	$p_4{=}1$, $d_4{=}2$		$p_3{=}2$, $d_3{=}10$		$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Sorted events	$\alpha{=}0.1$: $R{-}{=}1$ $P{+}{=}10$	$\alpha{=}0.2$: $R{+}{=}2$ $P{-}{=}10$	$\alpha{=}1/3$: $R{-}{=}1$ $P{+}{=}3$	$\alpha{=}0.5$: $R{-}{=}1$ $P{+}{=}2$	$\alpha{=}1$: $R{-}{=}1$ $P{+}{=}1$	$\alpha{=}5/3$: $R{+}{=}5$ $P{-}{=}3$	$\alpha{=}2$: $R{+}{=}2$ $P{-}{=}1$	$\alpha{=}5$: $R{+}{=}10$ $P{-}{=}2$
Prefix sum	$R{=}3$ $P{=}10$	$R{=}5$ $P{=}0$	$R{=}4$ $P{=}3$	$R{=}3$ $P{=}5$	$R{=}2$ $P{=}6$	$R{=}7$ $P{=}3$	$R{=}9$ $P{=}2$	$R{=}19$ $P{=}0$

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \le 1} 1 + \sum_{j:\alpha p_j \ge d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$

	$p_4{=}1$, $d_4{=}2$		$p_3{=}2$, $d_3{=}10$		$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Sorted events	$\alpha{=}0.1$: $R{-}{=}1$ $P{+}{=}10$	$\alpha{=}0.2$: $R{+}{=}2$ $P{-}{=}10$	$\alpha{=}1/3$: $R{-}{=}1$ $P{+}{=}3$	$\alpha{=}0.5$: $R{-}{=}1$ $P{+}{=}2$	$\alpha{=}1$: $R{-}{=}1$ $P{+}{=}1$	$\alpha{=}5/3$: $R{+}{=}5$ $P{-}{=}3$	$\alpha{=}2$: $R{+}{=}2$ $P{-}{=}1$	$\alpha{=}5$: $R{+}{=}10$ $P{-}{=}2$
Prefix sum	$R{=}3$ $P{=}10$ $\xi{=}3$	$R{=}5$ $P{=}0$ $\xi{=}2$	$R{=}4$ $P{=}3$ $\xi{=}2$	$R{=}3$ $P{=}5$ $\xi{=}1.5$	$R{=}2$ $P{=}6$ $\xi{=}-1$	$R{=}7$ $P{=}3$ $\xi{=}-5$	$R{=}9$ $P{=}2$ $\xi{=}-6$	$R{=}19$ $P{=}0$ $\xi{=}-12$

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha\, P$$

$$R = \sum_{j:\alpha p_j \le 1} 1 + \sum_{j:\alpha p_j \ge d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$

	$p_4{=}1$, $d_4{=}2$		$p_3{=}2$, $d_3{=}10$		$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$	
	j_4		j_3		j_1	j_2	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Sorted events	$\alpha{=}0.1$: $R{-}{=}1$, $P{+}{=}10$	$\alpha{=}0.2$: $R{+}{=}2$, $P{-}{=}10$	$\alpha{=}1/3$: $R{-}{=}1$, $P{+}{=}3$	$\alpha{=}0.5$: $R{-}{=}1$, $P{+}{=}2$	$\alpha{=}1$: $R{-}{=}1$, $P{+}{=}1$	$\alpha{=}5/3$: $R{+}{=}5$, $P{-}{=}3$	$\alpha{=}2$: $R{+}{=}2$, $P{-}{=}1$	$\alpha{=}5$: $R{+}{=}10$, $P{-}{=}2$
Prefix sum	$R{=}3$, $P{=}10$, $\xi{=}3$	$R{=}5$, $P{=}0$, $\xi{=}2$	$R{=}4$, $P{=}3$, $\xi{=}2$	$R{=}3$, $P{=}5$, $\xi{=}1.5$	$R{=}2$, $P{=}6$, $\xi{=}{-}1$	$R{=}7$, $P{=}3$, $\xi{=}{-}5$	$R{=}9$, $P{=}2$, $\xi{=}{-}6$	$R{=}19$, $P{=}0$, $\xi{=}{-}12$

Interpolation

Min>0: $\xi(0.5) = 1.5$

Max<0: $\xi(1) = -1$

$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \le 1} 1 + \sum_{j:\alpha p_j \ge d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$

	j_4		j_3		j_1	j_2	
	$p_4{=}1$, $d_4{=}2$		$p_3{=}2$, $d_3{=}10$		$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$	
Local job?	1	0	1	0	1	1	0
n	**4**	**4**	**4**	**4**	**4**	**4**	**4**

Sorted events	$\alpha{=}0.1$: $R{-}{=}1$ $P{+}{=}10$	$\alpha{=}0.2$: $R{+}{=}2$ $P{-}{=}10$	$\alpha{=}1/3$: $R{-}{=}1$ $P{+}{=}3$	$\alpha{=}0.5$: $R{-}{=}1$ $P{+}{=}2$	$\alpha{=}1$: $R{-}{=}1$ $P{+}{=}1$	$\alpha{=}5/3$: $R{+}{=}5$ $P{-}{=}3$	$\alpha{=}2$: $R{+}{=}2$ $P{-}{=}1$	$\alpha{=}5$: $R{+}{=}10$ $P{-}{=}2$
Prefix sum	$R{=}3$ $P{=}10$ $\xi{=}3$	$R{=}5$ $P{=}0$ $\xi{=}2$	$R{=}4$ $P{=}3$ $\xi{=}2$	$R{=}3$ $P{=}5$ $\xi{=}1.5$	$R{=}2$ $P{=}6$ $\xi{=}{-1}$	$R{=}7$ $P{=}3$ $\xi{=}{-5}$	$R{=}9$ $P{=}2$ $\xi{=}{-6}$	$R{=}19$ $P{=}0$ $\xi{=}{-12}$

Interpolation

Min>0: $\xi(0.5) = 1.5$

Max<0: $\xi(1) = -1$

$$\hat{\alpha} = 0.5 + \frac{1.5}{2.5} \cdot 0.5 = \mathbf{0.8}$$

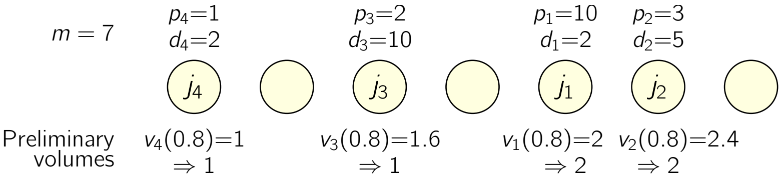
$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm



$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha\ P$$

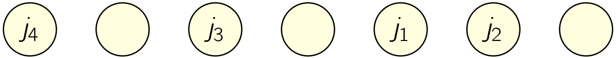
$$R = \sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

$m = 7$

	$p_4{=}1$, $d_4{=}2$	$p_3{=}2$, $d_3{=}10$	$p_1{=}10$, $d_1{=}2$	$p_2{=}3$, $d_2{=}5$



Preliminary volumes	$v_4(0.8){=}1$ $\Rightarrow 1$	$v_3(0.8){=}1.6$ $\Rightarrow 1$	$v_1(0.8){=}2$ $\Rightarrow 2$	$v_2(0.8){=}2.4$ $\Rightarrow 2$

Remaining volume: 1

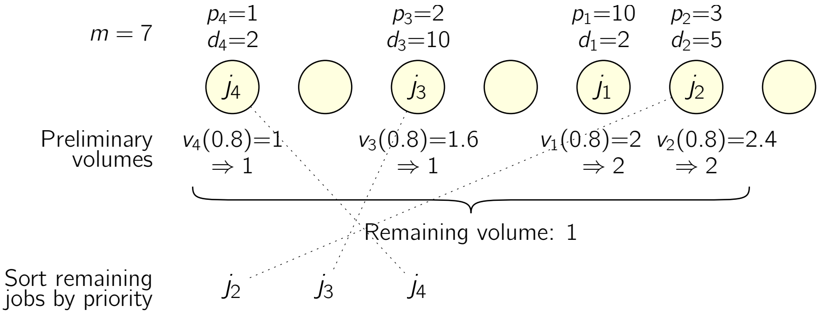
$$v_j(\alpha) \;=\; \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \textstyle\sum_j v_j(\alpha) = m - R - \alpha\, P$$

$$R = \textstyle\sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$

$$P = \textstyle\sum_{j:1<\alpha p_j<d_j} p_j$$

Volume Assignment: Algorithm

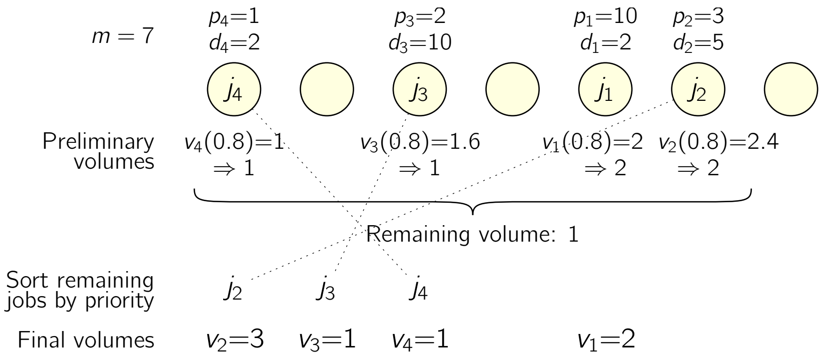


$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha)$$
$$= m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$
$$P = \sum_{j:1 < \alpha p_j < d_j} p_j$$

Volume Assignment: Algorithm



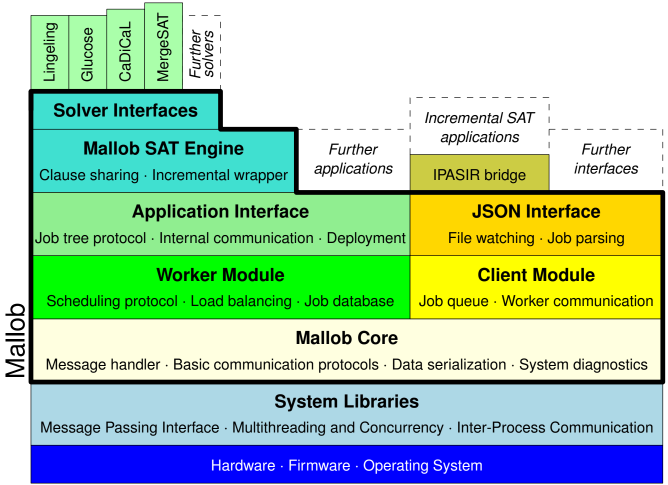
$$v_j(\alpha) = \max(1, \min(d_j, \alpha p_j))$$

$$\xi(\alpha) = m - \sum_j v_j(\alpha) = m - R - \alpha P$$

$$R = \sum_{j:\alpha p_j \leq 1} 1 + \sum_{j:\alpha p_j \geq d_j} d_j$$

$$P = \sum_{j:1<\alpha p_j<d_j} p_j$$

Mallob: Technology Stack



Realistic Arrival Rates: Worker Reuse

Worker reuse stategy	Workers created / Workers required		
	median	max.	total
None	1.43	33.0	2.14
Basic [Schreiber & Sanders, SAT'21]	1.40	31.5	2.07
Ours	**1.25**	**24.5**	**1.80**

Realistic Arrival Rates: Worker Reuse

Worker reuse stategy	Workers created / Workers required			Pr [Worker created at $\leq X$ processes]				
	median	max.	total	1	2	5	10	25
None	1.43	33.0	2.14	0.87	0.90	0.94	0.97	0.992
Basic [Schreiber & Sanders, SAT'21]	1.40	31.5	2.07	0.87	0.90	0.94	0.97	0.993
Ours	**1.25**	**24.5**	**1.80**	**0.89**	**0.91**	0.94	0.97	0.993

Realistic Arrival Rates: Worker Reuse

Worker reuse stategy	Workers created / Workers required			Pr [Worker created at $\leq X$ processes]				
	median	max.	total	1	2	5	10	25
None	1.43	33.0	2.14	0.87	0.90	0.94	0.97	0.992
Basic [Schreiber & Sanders, SAT'21]	1.40	31.5	2.07	0.87	0.90	0.94	0.97	0.993
Ours	**1.25**	**24.5**	**1.80**	**0.89**	**0.91**	0.94	0.97	0.993

$\Rightarrow$ Most workers ($\approx$ 90%) are initialized only once